03-04/2001

Linuxové noviny

Úvodem

David Häring

Druhé letošní vydání Linuxových novin vychází opět jako dvojčíslo. Redakční tým je bohužel o jednoho člověka chudší — tady přísluší poděkování Pavlu Juranovi, který dříve zajišťoval konverzi Linuxových novin do formátu HTML. Což ovšem rozhodně neznamená, že bychom od tohoto formátu chtěli upustit. Právě naopak — vynasnažíme se, aby HTML verze novin rychle následovala vydání v PDF verzi.

V tomto čísle se s Vámi Jiří Mlíka podělí o zkušenosti s používáním TV tuneru Askey TView99. Článek věnovaný problematice systémového logu se zabývá fungováním, konfigurací `syslogd` a také možnostmi zapisování do systémového logu z aplikací. Podíváme se na produkt Tripwire, určený ke kontrole integrity souborů a v článku o konfiguraci BINDu naleznete několik tipů, jak BIND provozovat bezpečněji. V minulém čísle jsme se zabývali monitorováním zátěže systému, tentokrát nás čeká monitorování dostupnosti služeb pomocí MONu. S novinkami, které nás čekají v novém GNU Emacsu nás seznámí Pavel Janík. Těm, kteří pracují na rozsáhlých projektech je určen rozsáhlý článek (vlastně je to spíše manuál) M. Ponkráce o CVS. Také se dočtete o kampani CZLUGu „Zdarma a legálně!“. Jako obvykle nechybí rubrika „Zasmáli jsme se“.

Příspěvky, náměty nebo i připomínky můžete posílat jako obvykle na adresu redakce (noviny@linux.cz). ■

Askey TView99

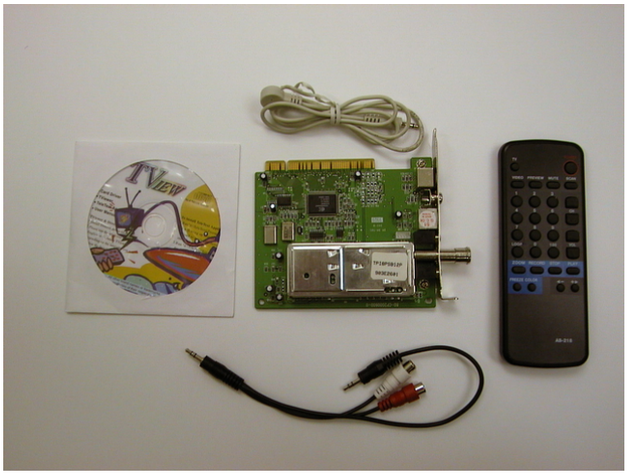
Jiří Mlíka, 16. března 2001

Obliba karet pro příjem televizního signálu mezi uživateli PC stále stoupá a to i mezi těmi, kteří na svém počítači používají Linux. V následujících řádcích vám jednu z nich představím. Jedná se o kartu Askey TView99, se kterou mám téměř roční zkušenost. TView99 patří do skupiny finančně méně náročných karet. Přesto je schopná dobře posloužit i nám, kteří si kromě práce na počítači najdeme čas i na sledování televize.

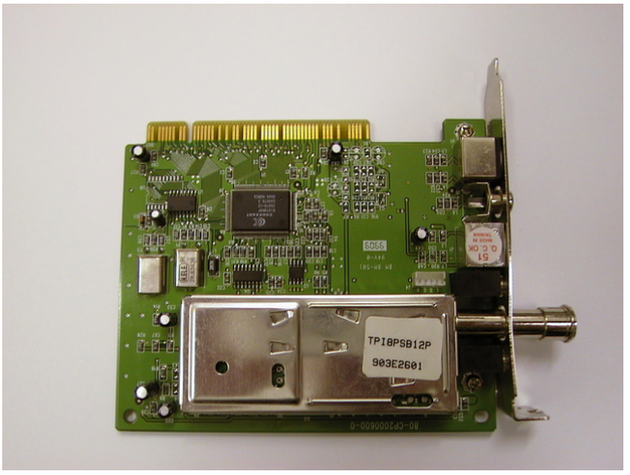
Po rozbalení krabice najdeme kromě samotné karty také dálkový ovladač a senzor dálkového ovládání, CD-ROM s ovladači, dokumentací, softwarem pro Windows a kabel na propojení se zvukovou kartou.

Karta se vyrábí v běžném provedení pro sběrnici PCI. Osazena je čipem BrookTree 878 společnosti Conexant, televizním tunerem pravděpodobně od Philipsu. Není bohužel vybavena čipem pro dekódování zvuku, jako je například msp4000, tzn. hraje pouze mono, což sice nepotěší, ale bohatě to pro běžné použití stačí. Zarytí audiofilové nechť se raději poohlédnou po něčem jiném.

Kromě klasického koaxiálního anténního vstupu lze použít i jiné zdroje video signálu. Na zadní straně najdete vstupní konektory pro S-Video, kompozitní video a vstup pro kameru v dokumentaci označovaný jako Philips Camera Input (CVBS). Mimo sledování televize lze tedy kartu použít například pro videokonference nebo nahrávání videoklipů do počítače.



Hardwarová část instalace je jednoduchá, spočívá v zasunutí karty do volného PCI slotu, máte-li ještě nějaký, a v propojení se zvukovou kartou pomocí dodaného kabelu. Kabel se zapojuje do linkového vstupu zvukové karty, vede tedy vnějškem počítače. Na kartě nalezneme i konektor pro propojení vnitřkem počítače, bohužel příslušný „drát“ budete v krabici hledat marně.



Zprovoznění karty po Linuxem

Přistupme tedy ke zprovoznění karty po Linuxem. Je celkem možné, že vaše distribuce kartu podporuje (vyzkoušeno s Mandrake 7.2), neboť autoři distribucí rádi upravují jádra tak, aby podporovala co nejvíce hardwaru. Pokud tomu tak není, nelze jinak než postupovat podle hesla „urob

Linuxové noviny připravuje a vydává České sdružení uživatelů operačního systému Linux ISSN 1213-3655

si sám“. Nejprve je třeba zajistit podporu tohoto zařízení v jádře. Toto se bude mírně lišit podle řady jádra, kterou používáte. Čip BrookTree 878 je podporován modulem bttv. U řady 2.4.x by mělo být vše bez problému. Řada 2.2.x (předpokládejme, že máte čisté nepatchované jádro) obsahuje zastaralou verzi ovladače (1), který Askey TView99 nepodporuje. Jsou dvě možnosti jak tento problém řešit. Buď naučit starého psa novým kouskům pomocí patchů, které najdete na serveru Hardware (2), nebo použít novější verzi bttv, která už TView99 podporuje a lze ji nalézt na stránkách autora. Pokud se rozhodnete pro druhou možnost, budete ještě potřebovat do jádra novou podporu pro **I2C** (3). Nejlepší způsob, jak vše zprovoznit, je přečíst si dokumentaci dodávanou k jednotlivým balíkům. Tomu se pravděpodobně stejně nevyhnete. Nebudu vše popisovat do detailů, pouze naznačím postup, který funguje na mém počítači s jádrem 2.2.16. Oba balíky nových ovladačů (jak **I2C** tak i **bttv**) nabízejí různé postupy zprovoznění. Vše je velice pěkně popsáno v dodávané dokumentaci. Osobně se mi osvědčilo použít I2C jako patch na jádro a bttv kompilovat samostatně mimo strom jádra.

Dodávaný dálkový ovladač je věc milá, bylo by škoda nechat jej zahálet. K jeho zprovoznění potřebujeme balík **LIRC** (4), který si sebou přináší potřebné „jaderné“ moduly pro různé typy IR zařízení a také potřebné démony pro práci s nimi. Zde bych chtěl upozornit, že takto zprovozněné dálkové ovládání lze využít nejen při sledování televize. Tlačítkům můžeme přiřadit libovolné akce jako je např. spouštění programů nebo jejich ovládání. Za povšimnutí stojí možnost používat dálkové ovládání jako myš jak na konzole tak pod X Window. Pod Red Hatem 7.0 se mi podařilo zkompilovat pouze LIRC verze 0.6.3pre4, jak mi poradil laskavý kolega z konference cz.comp.linux (5), tu ale musíte na domovské stránce projektu trošku více hledat.

Do souboru `/etc/modules.conf` můžete přidat například toto:

```
# i2c
alias char-major-89 i2c-dev
options i2c-core i2c_debug=1
options i2c-algo-bit bit_test=1
# bttv
alias char-major-81 videodev
alias char-major-81-0 bttv
options bttv pll=1 options tuner type=5
# LIRC
alias char-major-61 lirc_gpio
```

Pro hlavní účel, ke kterému byla karta stvořena, tedy sledování televize, budeme potřebovat nějakou vhodnou aplikaci rozumějící si s ovladačem bttv. Oblíbil jsem si dvě: XawTV (6) a KWinTV (7), pro čtení teletextu používám AleVT (8).

Nyní by snad bylo na místě říci něco ke kladům či záporům tohoto zařízení. To jak je obraz kvalitní a jak je možné jej doladit, závisí do značné míry na použité aplikaci a podmínkách příjmu televizního signálu. Těžko podat v tomto bodě nějakou objektivní informaci. Vycházím ze svých osobních zkušeností a konkrétních podmínek. Jako zdroj signálu používám běžnou společnou anténu v panelovém domě, obraz i zvuk je bez problémů. Těžko posoudit zda je to kvalitou použitého tuneru nebo sílou signálu v místě mého bydliště. Samozřejmě se nejedná o kvalitu špičkovou, což je pochopitelné uvědomíme-li si cenu zařízení. Tuner umožňuje jemné doladění, což bývá občas nutností.

Bohužel vše není tak růžové, jak by se na první pohled mohlo zdát. Existují uživatelé stěžující si na nepříjemné chrčení, objevující se při přístupu na harddisk či pohybu myši. Zajímavé je, že se to týká pouze některých uživatelů, ostatní si nestěžují. Ani po usilovném pátrání (ve kterém stále pokračuji) se mi nepodařilo zjistit příčinu. Není jasné zda se jedná o problém několika kusů nebo určitých hardwarových konfigurací. Ani komunikace s výrobcem nijak nepomohla.

Co říci závěrem?

Vzhledem k ceně a možnostem se jedná o příjemný doplněk pro uživatele PC, jenž dokáže zútulnit strohý pokoj na studentské koleji, nebo jako druhý televizní přijímač v domácnosti zabránit rodinným hádkám ohledně sledovaného programu. Obzvláště vhodný je pro „šťouravé“ uživatele, kteří se do sytosti „vyblbnou“ při konfiguraci dálkového ovládání.

Další užitečné odkazy čtenář nalezne na stránkách serveru tvfreak.cz (9). ■

1 bttv
http://www.strusel007.de/linux/bttv/
2 Server Hardware
http://hardware.penguin.cz/hwzaznam.php3?druh=153
3 I2C
http://www2.lm-sensors.nu/~lm78/
4 LIRC
http://www.lirc.org/
5 Konference cz.comp.linux
http://www.linux.cz/lists/
6 XawTV
http://www.strusel007.de/linux/xawtv/
7 KWinTV
http://www.mathematik.uni-kl.de/~wenk/kwintv/
8 AleVT
http://lecker.essen.de/~froese/
9 Stránky o TV kartách
http://www.tvfreak.cz

Jak na systémový log?

David Häring, 17. února 2001

V UNI*Xových systémech se o sběr, filtrování a ukládání systémových zpráv stará démon, který se zpravidla jmenuje `syslogd`. Zprávami zde rozumíme různá chybová, informační hlášení pocházející od jádra, systémových procesů, ale i aplikací. Obdobně tomu je i v linuxových distribucích. V distribuci Red Hat je `syslogd` součástí balíčku **sysklogd**, který kromě vlastního `syslogd` obsahuje i utilitu `klogd`, která funguje jako prostředník mezi `syslogd` a linuxovým jádrem, v distribuci Debian se používá syslog-ng.

Jak syslogd funguje

`Syslogd` čte příchozí zprávy ze schránky (socketu) `/dev/log` a na základě konfigurace, která se obvykle nachází v souboru `/etc/syslog.conf`, tyto zprávy filtruje a ukládá do příslušného souboru nebo souborů. V nejjednodušším případě může `syslogd` ukládat všechny zprávy do jednoho souboru, obvykle je ale žádoucí zprávy roztřídit podle jejich důležitosti či obsahu do určených souborů a zprávy nevýznamné „zapomenout“. Jedna zpráva také

může být zapsána na více výstupů: kritická zpráva může být například uložena do jednoho nebo více log souborů a ještě vypsána na konzoli.

Kategorie zpráv, priority

Každá zpráva kromě vlastního textu obsahuje dva atributy: prioritu a kategorii (angl. „facility“). Priorita říká, jak je daná zpráva významná. K dispozici je 9 úrovní priority, které jsou označeny následovně (seřazeno od nejnižší priority k nejvyšší): debug, info, notice, warning, err, crit, alert a emerg. Z důvodu zpětné kompatibility existují ještě synonyma warn (= warning), error (= err) a panic (= emerg), tyto by se ale již neměly používat.

Atribut kategorie říká, jaké oblasti se zpráva týká nebo od jaké služby pochází. K dispozici je 12 předdefinových kategorií:

- `auth` — autentizace, např. zprávy týkající se přihlašování / odhlašování uživatelů,
- `authpriv` — autentizace, vyhrazeno pro zprávy, které by z bezp. důvodů měly být odděleny od ostatních a které jsou určeny pouze administrátorovi systému,
- `cron` — zprávy od cronu — démona, který zajišťuje pravidelné spouštění akcí,
- `daemon` — blíže neurčené zprávy systémových aplikací,
- `kern` — zprávy jádra,
- `lpr` — zprávy týkající se tiskového systému (např. `lpd` apod.),
- `mail` — zprávy MTA (doručování pošty — např. `sendmail` apod.),
- `mark` — vyhrazeno pro tzv. „timestamps“; značky, které se periodicky zapisují do logu,
- `news` — zprávy NNTP serveru (diskusní skupiny — Usenet news),
- `security` — znamená totéž co „auth“, synonymum,
- `syslog` — zprávy syslogu,
- `user` — blíže neurčené zprávy uživ. aplikací,
- `uucp` — zprávy aplikací UUCP (Unix to Unix Copy Protocol, dnes se již téměř nepoužívá).

Existuje ještě 8 dalších kategorií určených k libovolnému použití. Tyto jsou označeny `local0` až `local7`.

Konfigurační soubor má jednoduchou syntaxi

```
{kategorie.[!=]priorita  místo_určení}
```

Pro označení kategorie i priority můžeme použít znak *, který má obdobně jako v regulárních výrazech význam „vyber vše“, tedy např. `kern.*` znamená všechny zprávy jádra bez ohledu na jejich prioritu. Jinak platí, že pokud prioritu uvedeme, pravidlu vyhoví zpráva dané kategorie s prioritou shodnou anebo vyšší (např. `mail.info` zahrnuje zprávy kategorie `mail` priority `info` a vyšší).

Pokud chceme vybrat pouze zprávy dané priority, použijeme operátor „=“ (např. `mail.=info`). Pokud chceme vybrat zprávy všech priorit s výjimkou jedné, použijeme operátor „!“ (např. `mail.*;mail.!=info` vybere zprávy kategorie `mail` všech priorit s výjimkou `info`). Pokud bychom chtěli vybrat zprávy kategorie `daemon` v rozsahu priorit `info` až `alert` použijeme následující zápis: `daemon.info;daemon.!alert`. U jednoho místa určení můžeme uvést i více kategorií zpráv, v tom případě je oddělíme čárkou, pokud se jedná o stejnou prioritu (např. `mail,daemon.info`), anebo středníkem, pokud jsou u jednotlivých kategorií priority různé (např. `kern.warning;daemon.info`).

Místo určení zpráv může být buď cesta k log souboru, cesta k pojmenované rouře uvozená znakem „|“ (roura musí existovat před spuštěním syslogu), cesta k terminálovému zařízení (např. `/dev/console`), jméno vzdáleného systému uvozené znakem „@“ (např `@server.domena.cz`), seznam uživatelů oddělených čárkou (zpráva jim bude vypsána na terminál, pokud jsou v okamžiku obdržení zprávy syslogem přihlášeni v systému) a konečně, pokud chceme aby zprávu obdrželi na terminál všichni momentálně přihlášení uživatelé, použijeme znak „*“. Pokud uvádíme více kategorií zpráv, oddělíme je středníkem. Řádky uvozené znakem „#“ jsou považovány za komentáře.

Logování přes síť

Jak už bylo dříve zmíněno, je možné systémový log prostřednictvím sítě zapisovat i na jiný stroj. Můžeme tedy mít např. počítač, který centrálně sbírá a zapisuje logy z ostatních počítačů. Klasický syslog komunikuje přes protokol UDP na portu 514 (viz `/etc/services`). Na stroji, odkud chceme logy přeposílat na vzdálený systém stačí v konfiguračním souboru uvést jméno vzdáleného systému (např. `@server.domena.cz`). Na vzdáleném systému, který má log přijímat, je potřeba `syslogd` spustit s volbou `-r`. Je potřeba si uvědomit, že `syslogd` nemá žádný mechanismus pro omezení přístupu ke vzdálenému logování, čehož lze snadno využít k zahlcení či zaplnění diskového prostoru (DoS, útok způsobující odepření služby). Proto by měl být stroj, který logování přes síť umožňuje, chráněn např. pomocí firewallu tak, aby umožnil spojení na UDP port 514 pouze z těch strojů, která jsou k tomu oprávněny.

Logování z aplikací

Knihovna libc poskytuje pro zápis do systémového logu jednoduché rozhraní (viz ukázka zápisu do systémového logu z C). Sestává z funkcí `openlog()`, `syslog()` a `closelog()`. Nejprve při volání funkce `openlog()` zvolíme kategorii zpráv. Zápisy do systémového logu pak provádíme voláním funkce `syslog()`, které předáváme jako parametry prioritu zprávy a její text.

Logování z příkazové řádky

Pokud potřebujeme zapisovat do systémového logu ze skriptu, můžeme použít utilitu `logger` z balíčku **util-linux**. Například příkaz:

```
server$ logger -i -t muj_skript \
 -p daemon.info "test mesage"
```

do systémového logu zapíše:

```
Feb 10 17:05:44 server muj_skript[16576]: \
 test mesage
```


```
#include <syslog.h>

#define LOG_FACILITY LOG_LOCAL0

int main (int argc, char ** argv)
{
  /* nastavíme způsob logování
  - zvolíme kategorii zpráv */
  openlog(argv[0],LOG_PID|LOG_CONS,LOG_FACILITY);

  /* zápis do logu */
  syslog(LOG_INFO,"zapisujeme do logu.");

  closelog();
}
```

Výpis č. 1: ukázka zápisu do systémového logu z C

Syslogd a logování aplikací v chroot prostředí

Syslogd obvykle čte ze schránky (socketu) /dev/log. Pokud máme aplikaci, která zapisuje do systémového logu a kterou provozujeme v *chroot* prostředí (např. bind), potřebujeme této aplikaci nějak zpřístupnit schránku syslogu. Novější verze syslogd umí pracovat s více schránkami (zadává se pomocí volby -a). Pokud máme starší verzi syslogd, můžeme použít utilitu holelogd (1), která otevře schránku na požadovaném místě a příchozí zprávy zapisuje do /dev/log.

Rotace logů

Rotace logů je zajišťována externím programem, což bývá nejčastěji logrotate. Ten čte konfiguraci ze souboru /etc/logrotate.conf případně ze souborů v adresáři /etc/logrotate.d. Je možné zadat jak často nebo při jaké velikosti se mají log soubory obměňovat, jak dlouho se mají logy uchovávat, případně jestli se mají starší logy komprimovat.

Výhody, nevýhody a alternativy

Slabou stránkou klasického syslogd je zejména logování přes síť a neexistující ochrana před neoprávněnou modifikací log souborů. Kromě toho, že neobsahuje žádnou kontrolu přístupu a tím je zneužitelný k útokům odepřením služby (DoS), **syslogd** komunikuje přes protokol UDP, což znesnadňuje zapouzdření komunikace přes bezpečný kanál (mějme na paměti, že systémový log obsahuje citlivá data, která by neměla procházet přes nedůvěryhodné médium). Alternativou ke klasickému syslogd je třeba syslog-ng (2), který umí komunikovat jak přes UDP tak i přes TCP a který navíc umožňuje flexibilnější filtrování zpráv pomocí regulárních výrazů. Dalšími alternativami, s důrazem na bezpečnost je třeba Secure syslog (3) nebo novější Modular syslog (4), který nabízí mimo jiné ukládání logu do databází, filtrování zpráv pomocí regulárních výrazů a ochranu proti manipulaci s logy. Dalším zajímavým produktem je Nsyslogd (5), který podporuje komunikaci přes TCP/SSL. Na tyto alternativy klasického syslogu se podíváme příště. ■

1 holelogd
ftp://ftp.obtuse.com/pub/utils

2 syslog-ng
http://www.balabit.hu/en/products/syslog-ng/
3 Secure syslog
http://www.core-sdi.com/english/freesoft.htm
4 Modular syslog
http://www.core-sdi.com/english/slogging/modular-dl.htm
5 Nsyslogd
http://coombs.anu.edu.au/~avalon/nsyslog.html

Kontrola integrity systému — Tripwire

David Häring, 3. března 2001

Tripwire je software určený pro kontrolu integrity souborů v systému. Jak takový systém funguje? Nejprve si sestaví databázi, která popisuje stav zadaných souborů, adresářů nebo celých adresářových stromů. S touto databází je pak možné kdykoliv srovnat aktuální stav a identifikovat případné změny, které mohou zahrnovat přidané či smazané objekty, změny přístupových práv, modifikace souborů a podobně. Administrátor pak má možnost změny autorizovat a tím změny zanést do databáze, takže příští kontrola vyjde z aktualizované databáze.

Tripwire samozřejmě není zdaleka jediným nástrojem svého druhu, je ovšem jedním z prvních a verze 2.3.0 pro Linux je od nedávna dostupná včetně zdrojových kódů (1). Pro ostatní systémy je volně dostupná starší verze 1.3.1 formou zdrojových kódů jako ASR (Academic Source Release). Z alternativ za zmínku rozhodně stojí např. AIDE (2), řadu dalších alternativ čtenáři naleznou např. na serveru Freshmeat (3).

Společnost Tripwire také nabízí řadu dalších produktů s rozšířenou funkčností oproti základní verzi a to Tripwire Manager (GUI nástroj pro vzdálenou administraci) a Tripwire for Servers (může být spravován dálkově Tripwire Managerem). Posledním produktem je Tripwire for Web Pages, který spolupracuje s web serverem (apache) — funguje tak, že kontroluje odesílané HTML soubory a pokud narazí na stránku, která se změnila (a tato změna nebyla autorizována), tak místo ní odešle stránku s hlášením o dočasné nedostupnosti dané stránky a uvědomí administrátora. V tomto článku se budeme zabývat jen volně dostupnou verzí Tripwire pro Linux.

Co nabízí Tripwire

Jaké atributy tripwire u jednotlivých souborů sleduje? V systémech UN*Xového typu (tedy i Linuxu) jsou dostupné následující atributy, ze kterých je možno zvolit libovolnou kombinaci:

- přístupová práva
- číslo i-uzlu
- počet odkazů na soubor
- UID (ID uživatele — vlastníka souboru)
- GID (ID skupiny — vlastníka souboru)
- typ souboru
- velikost souboru
- velikost souboru pouze roste (log soubory apod.)
- číslo blokového zařízení, na kterém se soubor nalézá

```
Tripwire(R) 2.2.1 Integrity Check Report

Report generated by:          root
Report created on:            Sat Mar  3 16:07:38 2001
Database last updated on:     Fri Mar  2 13:46:35 2001

===============================================================================
Report Summary:
===============================================================================

Host name:                    server.domena.cz
Host IP address:              11.22.33.44
Host ID:                      b2df4h54
Policy file used:             /usr/tripwire/policy/tw.pol
Configuration file used:      /usr/tripwire/bin/tw.cfg
Database file used:           /usr/tripwire/db/server.twd
Command line used:            bin/tripwire -m c -I

===============================================================================
Rule Summary:
===============================================================================

-------------------------------------------------------------------------------
  Section: Unix File System
-------------------------------------------------------------------------------

  Rule Name                     Severity Level    Added    Removed  Modified
  ---------                     --------------    -----    -------  --------
  Invariant Directories         66                0        0        0
  Tripwire Data Files           100               0        0        0
  Temporary directories         33                0        0        0
  Critical devices              100               0        0        0
  Tripwire Binaries             100               0        0        0
  User binaries                 66                0        0        0
  setuid/setgid                 100               0        0        0
  Diskless executables/libraries 100              0        0        0
  Diskless config files         100               0        0        0
  Libraries                     66                0        0        0
  OS executables and libraries  100               0        0        0
  Shell Binaries                66                0        0        0
* Critical configuration files  100               0        0        1
  Security Control              66                0        0        0
  Boot Scripts                  66                0        0        0
  Login Scripts                 66                0        0        0
  System boot changes           100               0        0        0
  Critical system boot files    100               0        0        0
  Root config files             100               0        0        0

Total objects scanned:  27070
Total violations found:  1
```

Výpis č. 2: Ukázka výsledné zprávy tripwire

- čísla zařízení u speciálních souborů
- počet alokovaných datových bloků souboru
- čas modifikace souboru
- čas modifikace i-uzlu
- čas přístupu

Dále jsou k dispozici 4 typy kontrolních součtů, které slouží ověření neporušenosti obsahu souborů:

- CRC-32 (nízká bezpečnost, rychlý)
- MD5 (vyšší bezpečnost)
- SHA (vyšší bezpečnost)
- HAVAL (128 bitový klíč, nejvyšší bezpečnost)

Na jeden soubor můžeme aplikovat libovolnou kombinaci kontrolních součtů současně, pouze je třeba vzít v úvahu, že pokud jich použijeme více, čas potřebný pro kontrolu poroste. Při volbě kontrolních součtů tedy volíme kompromis v závislosti na množství dat, které potřebujeme prověřovat, stupni bezpečnosti a času, za jaký musí kontrola proběhnout.

```
================================================================================
Object Detail:
================================================================================

--------------------------------------------------------------------------------
  Section: Unix File System
--------------------------------------------------------------------------------

--------------------------------------------------------------------------------
Rule Name: Critical configuration files (/etc/hosts.allow)
Severity Level: 100
--------------------------------------------------------------------------------
  ---------------------------------------
  Modified Objects: 1
  ---------------------------------------

Modified object name:  /etc/hosts.allow

  Property:            Expected                  Observed
  -------------        -----------               -----------
* Size                 1047                      1065
* Modify Time          Sun Feb  4 11:12:32 2001  Sat Mar  1 15:09:53 2001
* Change Time          Sun Feb  4 11:12:32 2001  Sat Mar  1 15:09:53 2001
* CRC32                Cx9OxW                    BNKusC
* MD5                  Codzk4vzD9QR2gXx7QSwRU    CaHexEYPYLqcqMqz2Fj2ky

================================================================================
Error Report:
================================================================================

--------------------------------------------------------------------------------
  Section: Unix File System
--------------------------------------------------------------------------------

No Errors
--------------------------------------------------------------------------------
*** End of report ***

Copyright (C) 1998-2000 Tripwire(R) Security Systems, Inc.
Tripwire(R) is a~registered trademark of the Purdue Research
Foundation and is licensed exclusively to Tripwire(R) Security
Systems, Inc.
```

Výpis č. 2: Ukázka výsledné zprávy tripwire (pokračování)

Jednotlivé soubory je možné sdružovat do celků, kterým lze přiřazovat atributy jako je významnost („severity“, při kontrolách pak volíme jak významné změny nás zajímají), dále emailová adresa, na kterou se posílá upozornění v případě zjištění změny, příznak určující jestli se má daný adresář procházet rekurzívně. Můžeme tedy například jednoduše říci, že chceme sledovat všechny soubory a podadresáře v adresářovém stromu `/etc`, přiřadit souborům v stromu `/etc` maximální význam a e-mailovou adresu, čímž zajistíme, že v případě jakékoliv změny detekované při kontrole budou administrátoři upozorněni e-mailem.

Instalace

K dispozici je buď původní verze Tripwire 2.2.1 (4) ve formě `tar.gz` archívu s instalačním skriptem, nebo verze 2.3 ve formě RPM balíčku (5), případně si zájemci mohou stáhnout aktuální zdrojový kód verze 2.3.1 (6) a binárky sestavit sami. Rozdíly mezi uvedenými verzemi jsou minimální, verze 2.3 obsahuje drobnou opravu při manipulaci s dočasnými pracovními soubory, konfigurační soubory tripwire jsou pak na rozdíl od dřívějších verzí standardně ukládány v adresáři `/etc/tripwire`. Ve verzi 2.3.1 přibyla navíc podpora platformy FreeBSD. Instalační skript původní verze 2.2.1 automaticky nerozpoznává novější linuxové distribuce (což ale nijak nevadí, protože to nijak neovlivní funkčnost instalace).

V průběhu instalace se generují klíče, ke kterým je třeba zadat globální („site passphrase“) a lokální heslo („local passphrase“). Obě hesla jsou důležitá; globální heslo je později vyžadováno při úpravách bezpečnostní politiky, lokální heslo je vyžadováno při aktualizaci databáze.

Konfigurace, inicializace databáze

Konfigurační soubor `tw.cfg` obsahuje základní údaje o in-

stalaci, t.j. cestu k instalaci tripwire, cestu k souboru s definicí bezpečnostní politiky, cesty ke klíčům, nastavení implicitního editoru, nastavení způsobu odesílání pošty apod. Pokud chceme konfiguraci prohlížet nebo ji měnit, musíme ji utilitou `twadmin` převést do textové podoby (`twadmin -m f`) a po ukončení editace opět utilitou `twadmin` zašifrovat (`twadmin -m F`).

Soubor `tw.pol` obsahuje definici vlastní bezpečnostní politiky, t.j. soubor pravidel udávajících co a jakým způsobem se má kontrolovat. Po instalaci tripwire je k dispozici implicitní soubor pravidel, který je potřeba upravit podle vlastních potřeb. Stejně jako konfigurační soubor `tw.cfg` i soubor s politikou je ukládán v zašifrované podobě a před editací je třeba je převést do textové podoby utilitou `twadmin` (`twadmin -p`) a po editaci zašifrovat (`twadmin -P`). Součástí instalace je soubor `policyguide.txt`, kde je na řadě příkladů a tipů v praxi vysvětlena syntaxe konfiguračního souboru bezpečnostní politiky.

Jakmile máme konfiguraci hotovu, můžeme vytvořit databázi spuštěním příkazu `tripwire --init`.

Kontrola systému, prohlížení zpráv

Funkčnost instalace ověříme spuštěním kontroly systému. Máme na výběr ze dvou možností, a to buď interaktivní anebo „offline“ režim. V interaktivním režimu (`tripwire -m c -I`) tripwire po skončení spustí editor (implicitně `vi`), do kterého natáhne detailní výslednou zprávu. Ve zprávě pak můžeme odsouhlasit některé nebo i všechny změny a po ukončení práce s editorem tripwire provede aktualizaci databáze podle odsouhlasených změn. Výsledná zpráva je rovněž uložena do příslušného adresáře (implicitně `/var/lib/tripwire/report`) a také může být zaslána elektronickou poštou, je-li to odpovídajícím způsobem uvedeno v bezpečnostní politice (direktiva `emailto`). Vygenerované zprávy lze kdykoliv prohlížet pomocí utility `twprint` (viz Ukázka výsledné zprávy tripwire).

Aktualizace bezpečnostní politiky

Pokud provádíme změny bezpečnostní politiky, neměli bychom tak činit prostým nahrazením starého souboru s bezpečnostní politikou novým a opětovnou inicializací databáze, ale použít režimu tripwire pro aktualizaci politiky (`tripwire -m p`). Tripwire pak zkontroluje systém podle staré i nové bezpečnostní politiky a teprve v případě že je systém v pořádku podle obou verzí bezpečnostní politiky provede aktualizaci databáze a souboru s bezpečnostní politikou.

Zásady práce s tripwire

Samozřejmě, pouhé instalování tripwire (nebo jakéhokoliv jiného systému kontroly integrity) bez pečlivé konfigurace a dodržování rozumných zásad úroveň zabezpečení systému nezvýší, pouze povede k falešnému pocitu „bezpečí“. Jaké zásady by tedy měl administrátor při nasazení takového systému pro kontrolu integrity dodržovat? Minimálně alespoň databázové soubory je potřeba uchovávat odděleně na důvěryhodném médiu, nejlépe digitálně podepsané. Optimální je mít na odděleném médiu i binárky tripwire (třeba jako doplňek k „záchranné“ sadě bootovacích disket) a kontrolu provádět po nabootování z důvěryhodného média. Tripwire lze samozřejmě spouštět i pravidelně třeba z cronu, ale i pak je potřeba provádět občasné kontroly manuálně z důvěryhodných médií. Také je potřeba provádět aktualizace databáze v návaznosti na prováděné zásahy do systému. Rozhodně např. nemá smysl aktualizovat databázi koncem týdne a přitom přemýšlet, které změny provedl v průběhu týdne který z pěti administrátorů.

Závěrem

Tripwire je kvalitním nástrojem pro kontrolu integrity souborů a pokud je dobře nakonfigurován, výrazně může pomoci nejen při detekci neautorizovaných zásahů do systému, ale i při odstraňování následků bezpečnostního incidentu. Pokud ještě žádný takový systém nepoužíváte, rozhodně Tripwire (anebo některou z jeho alternativ) vyzkoušejte. ■

1 Open Source verze tripwire
http://www.tripwire.org/
2 AIDE
http://www.cs.tut.fi/~rammer/aide.html
3 Freshmeat
www.freshmeat.net
4 Tripwire 2.2.1
http://www.tripwire.com/downloads/tripwire_221/
5 RPM balíček tripwire
http://www.tripwire.org/downloads/index.php
6 Zdrojový kód Tripwire verze 2.3.1
http://sourceforge.net/projects/tripwire/

BIND a bezpečnost?

David Häring, 25. února 2001

Bind je bezesporu nejrozšířenějším softwarem implementujícím službu DNS (Domain Name System, základy implementace DNS nalezneme např. v RFC 1034 a 1035). Také není pochyb o tom, že dlouhodobě obsazuje nejvyšší příčky v nepopulárních žebříčcích aplikací, které obsahují závažné bezpečnostní chyby a jejichž provozování má ve svém důsledku na svědomí nemalou část úspěšných průniků do systémů nebo útoků způsobujících znepřístupnění služby (DoS, tzv. „Denial of Service“ útok). Zároveň ovšem dodejme, že k této situaci přispívá v nemalé míře fakt, že mnoho administrátorů podceňuje rizika spojená s provozováním nejen bindu na systémech přístupných z Internetu a že nevěnují dostatečnou pozornost opravám, které pravidelně vycházejí. Dokladem budiž například starší článek na serveru underground.cz (1). Tento článek, který mapoval použití různých verzí bindu v doméně .cz sice vyšel více než před půl rokem, nicméně ukázal, že i někteří provozovatelé frekventovaných nameserverů jsou s opravami pozadu. Když pak během ledna letošního roku bylo publikováno několik dalších chyb v bindu, následovalo opět období, kdy se konference hemžily zprávami o úspěšných průnicích, přitom opravené verze bindu byly na webu ISC k dispozici poměrně rychle po publikování chyb. Přehled verzí bindu (2) s popisem chyb, které obsahují je k dispozici na www stránkách ISC.

Jaké tedy má administrátor provozující name server možnosti, chce-li předejít bezpečnostnímu incidentu? Příznivci DJB (3) (4) navrhnou jako alternativu použít **djb-dns** (5). Podrobně se instalací djb-dns zabývali např. na serveru Root.cz (6). Djb-dns nepochybně je kvalitní náhradou

bindu, má ovšem také zcela odlišnou architekturu a filozofii. Co tedy zbývá administrátorům, kteří na djb-dns přejít nemohou anebo nechtějí? Další možností by bylo přejít na bind verze 9.1, který je ovšem stále poměrně mladý. V tomto článku se zaměříme na několik zásad, které se týkají bezpečnosti instalace a konfigurace bindu verze 8.2.3-REL, což je v době sestavování tohoto článku poslední stabilní verze řady 8.2.

Bezpečnější instalace — chroot, uid, gid

Bind obsahuje několik mechanismů pro zajištění bezpečnosti: umí běžet v *chroot* prostředí, umožňuje volbu uživatele a skupiny, pod kterými poběží. Tyto možnosti bychom měli v každém případě využít. V řadě standardních instalací linuxových distribucí těchto možností nebývalo využito, což pak vedlo ke snadnému kompromitování celého systému v případě úspěšného útoku.

Chroot prostředí v podstatě znamená, že určitá aplikace je uzamčena v určité vyhrazené části souborového systému, kterou považuje za kořenový adresář a mimo něj nevidí. V případě, že dojde k bezpečnostnímu incidentu, útočník může přistupovat pouze k datům v rámci adresářové struktury, která je přístupná v rámci daného prostředí *chroot* a možnost infiltrace ostatních částí systému je mizivá. Je samozřejmé, že aplikace přitom nesmí běžet s oprávněním uživatele či skupiny root. (Uživatel root může např. pomocí volání jádra `mknod` vytvořit potřebná speciální zařízení, přes která pak může přistupovat k diskům apod.)

Existuje několik způsobů, jak libovolnou aplikaci spouštět v *chroot* prostředí. Jednou z možností je použít volání jádra `chroot` v dané aplikaci (detaily použití viz manuálová stránka), což ovšem vyžaduje úpravu zdrojového kódu aplikace. Druhou možností je použít nějaký „wrapper“, např. utilitu `chroot` z balíčku **sh-utils**. Bind ovšem s možností běhu v prostředí *chroot* počítá, a proto není potřeba do něj zasahovat, stačí vše patřičně nakonfigurovat a spustit jej s příslušnými volbami (`-t`, `-g` a `-u`).

Dejme tomu, že chceme name server provozovat v adresáři `/bindroot` a dále, že bind poběží pod za tímto účelem vytvořeným uživatelem `dns` ve skupině `dns`. Aby to mohlo fungovat, je potřeba aby v adresáři `/bindroot` byly obsažena veškerá data a adresářové struktury, které bind ke svému běhu potřebuje. Rovněž je třeba upravit přístupová práva tak, aby s daty mohl uživatel dns pracovat (t.j. např. nastavit rekurzívně v adresáři `/bindroot/var/named`, kde budou umístěna data týkající se jednotlivých zón, vlastníka na uživatele a skupinu dns, dále named potřebuje zapsat PID do souboru `/bindroot/var/run/named.pid` a pokud k logování nepoužijeme syslog, potřebuje práva k zápisu do adresáře, kde jsou logy umístěny).

```
/bindroot
|-- dev
|   +-- null
|-- etc
|   |-- group
|   |-- localtime
|   |-- named.conf
|   +-- passwd
|-- usr
|   +-- sbin
|       |-- named
|       +-- named-xfer
+-- var
    |-- log
    |   |-- named.log
    |   +-- named_stats.log
    |-- named
    |   |-- d.domena.cz
    |   |-- d.reverz-domena.cz
    |   |-- named.ca
    |   +-- secondary
    |       +-- d.domena2.cz
    +-- run
        |-- named.pid
        +-- ndc
```

Soubory `passwd` a `group` v prostředí *chroot* by měly pochopitelně obsahovat pouze záznam o uživateli a skupině dns. Na uživatelský účet bindu by také mělo být zakázáno přihlašování se jakýmkoliv způsobem.

Proč používat acl?

Bind umí pracovat s ACL seznamy (Access Control List), takže není problém upravit přístup k jednotlivým zónám nebo typům údajů na základě IP adres. Tuto možnost bychom rozhodně neměli podceňovat, protože řada bezpečnostních chyb v bindu se týkala okrajových situací, kterým lze zabránit správnou konfigurací.

Konfigurace acl

Dejme tomu, že náš name server slouží jednak jako primární pro doménu „domena.cz“, sekundární pro doménu domena2.cz a dále slouží jako „caching“ name server pro firemní síť, která se skládá z počítačů 11.22.33.41 až 45. Dále předpokládejme, že sekundární name server pro naši doménu domena.cz je na stroji ns.nekdejinde.cz s IP adresou 22.33.44.55.

Definujeme dva ACL seznamy: seznam „trusted_query“, který obsahuje IP adresy firemních pracovních stanic pro které fungujeme jako „caching“ name server. Dále definujeme druhý seznam „trusted_axfr“, který bude obsahovat počítače, které smí provádět transfer zón — tento bude zahrnovat stroj, který funguje jako sekundární server pro doménu domena.cz a dále bude zahrnovat stroje CZ NICu, ze kterých jsou prováděny kontroly nastavení DNS. (Jména a adresy těchto strojů na požádaní CZ NIC sdělí, zde v příkladu uvedený seznam by měl být dostačující).

```
// seznam strojů kterým je povolen transfer zón
acl trusted_axfr {
// sekundární nameserver
22.33.44.55    // ns.nekdejinde.cz;
// CZ-NIC
193.85.1.12;   // ns.eunet.cz
193.85.3.130;  // cz.eunet.cz
193.85.7.100;  // ns.o.cz
};

// seznam strojů které se mohou dotazovat
// bez omezení
acl trusted_query {
localhost;
// nase stroje
11.22.33.41
11.22.33.42
```

```
case "$1" in
  start)
        echo -n "Starting system logger: "
        # otevřeme další socket v /bindroot/dev/log
        daemon syslogd -m 0 -a /bindroot/dev/log
        # anebo pokud používáme holelogd použijeme následující řádek
        # daemon /usr/local/sbin/holelogd /bindroot/dev/log
        RETVAL=$?
        echo
        echo -n "Starting kernel logger: "
        daemon klogd
        echo
        [ $RETVAL -eq 0 ] && touch /var/lock/subsys/syslog
        ;;
```

Výpis č. 3: ukázka části souboru /etc/rc.d/init.d/syslog

```
11.22.33.43
11.22.33.44
11.22.33.45
};
```

U direktivy options pak v konfiguračním souboru uvedeme, že transfer zón mohou provádět pouze stroje ze seznamu „trusted_axfr“ a standardně dotazy omezíme pouze na stroje ze seznamu „trusted_query“:

```
options {
        directory "/var/named";
        pid-file "/var/run/named.pid";
        allow-transfer { trusted_axfr; };
        allow-query { trusted_query; };
        allow-recursion { trusted_query; };
};
```

U deklarací jednotlivých zón, pro které je náš nameserver autoritativní a na dotazy, v rámci nichž musí server odpovídat všem bez rozdílu, pak v konfiguračním souboru explicitně dotazy zvenčí povolíme direktivou allow-query:

```
// master zóna
zone "domena.cz" {
        type master;
        file "d.domena.cz";
        allow-update   { none; };
        allow-query { any; };
};

// sekundární zóna
zone "domena2.cz" {
        type slave;
        file "secondary/d.domena2.cz";
        allow-query { any; };
        masters {
                22.33.44.55;
        };
};
```

Možností jak zabezpečit vlastní přenosy zón je použití autentizace serverů při přenosu zón pomocí podpisů (TSIG, transaction signatures). K tomu je potřeba vygenerovat klíč utilitou dns-keygen. Takto např. vygenerujeme do souboru Ktransferkey* MD5 klíč o délce 128 bitů:

```
server$ dnskeygen -H 128 -h -n transferkey.
Generating 128 bit HMAC-MD5 Key for transferkey.
Generated 128 bit Key for transferkey. \
id=0 alg=157 flags=513
```

V konfiguračním souboru pak přiřadíme klíče jednotlivým serverům, které jej budou používat:

```
key transferkey {
        algorithm hmac-md5;
        secret "srJPgeDYD42yitdbexG3Vg==";
};

server 22.33.44.55 {
        keys { transferkey ; };
};
```

Logování

Aplikace do systémového logu zapisují přes schránku (socket), který je zpravidla umístěn v /dev/log. Jestliže provozujeme aplikaci v *chroot* prostředí, musíme zajistit způsob, jakým komunikaci aplikace se syslogd zprostředkujeme. Poslední verze syslogd mohou mít otevřených více schránek. Pokud používáme klasický syslogd, řešením je spustit jej s volbou -a s uvedením cesty k další schránce (viz ukázka části souboru /etc/rc.d/init.d/syslog v distribuci Red Hat). Pokud to verze syslogd neumožňuje, můžeme použít utilitu holelogd (7). Ta funguje tak, že otevře schránku na určeném místě a příchozí zprávy kopíruje do standardní schránky syslogu v /dev/log.

Pokud chceme zprávy bindu oddělit od zpráv ostatních démonů, můžeme v konfiguračním souboru bindu zvolit jakou kategorii zpráv („facility“) a prioritu má bind pro zápis do syslogu použít.

```
logging {
        channel eventlog {
                syslog local5;
                severity info;
        };
        category default { eventlog; };
};
```

Další možností je syslog nepoužívat vůbec a nechat logovat přímo bind. Rovněž je možné potlačit vypisování některých druhů zpráv, anebo je třeba zapisovat odděleně do několika souborů. Následující příklad ukazuje konfiguraci, při které budou statistiky zapisovány odděleně od ostatních

zpráv. Zprávy některých kategorií, které informují o chybách v konfiguraci cizích domén budou ignorovány.

```
logging {

        channel statlog {
                file "/var/log/named_stats.log";
                severity info;
                print-time yes;
        };

        channel eventlog {
                file "/var/log/named.log";
                severity info;
                print-time yes;
                print-category yes;
        };

        category default { eventlog; };
        category statistics { statlog; };
        category lame-servers { null; };
        category cname { null; };
};
```

Instalace

Samozřejmě existují i RPM balíčky (8) s bindem připraveným pro běh v *chroot* prostředí. Další možností je podle výše popsaných zásad vytvořit *chroot* instalaci buď ze stávající instalace bindu anebo vyjít přímo ze zdrojových kódů.

Pokud instalujeme bind ze zdrojových kódů, můžeme se rozhodnout, jestli chceme, aby výsledný spustitelný soubor byl linkován staticky anebo dynamicky. Přehlednější je kompilovat staticky, protože instalujeme-li dynamicky linkovanou aplikaci do prostředí *chroot*, musíme tam také doplnit sdílené knihovny, které aplikace k běhu potřebuje (utilita `ldd` vypíše seznam knihoven, které daná aplikace vyžaduje). Implicitně se sestaví bind linkovaný dynamicky, pokud to chceme změnit, musíme před kompilací upravit soubor `Makefile.set` v adresáři `src/port/linux` (nastavení proměnné `CDEBUG`, např. `CDEBUG=-O2 -g -static`).

Poznámka ke kompilaci: v posledních distribucích zpravidla nelze bind staticky přeložit kvůli konfliktu symbolů stejného jména v bindu a glibc. Pomůže upravit první řádek v souboru `Makefile.set` např. následujícím způsobem:

```
CC=gcc -D_GNU_SOURCE\
 -D__res_randomid=__res_randomidmy
```

Shrnutí

Cílem tohoto článku bylo ukázat na možnosti konfigurace bindu, které velmi ovlivňují bezpečnost celého systému, na kterém je bind provozován. Vzhledem k tomu, že nástroje pro detekci a využití bezpečnostních děr v bindu jsou velmi populární a rozšířené, je v podstatě nemyslitelné jej provozovat jinak než v *chroot* prostředí pod zvláštním uživatelem a skupinou a co nejvíce jej tak oddělit od ostatních částí systému. Rovněž používání ACL seznamů pro omezení přístupu k jednotlivým funkcím či druhům informací, které bind poskytuje, lze rozhodně doporučit. Další informace, týkající se provozování bindu v chroot prostředí, či tipy na zabezpečení instalací lze nalézt např. v následujících odkazech: (9) (10) (11). ■

1 Malý český průzkum velkého bindu
http://underground.cz/453
2 Přehled verzí bindu
http://www.isc.org/products/BIND/bind-security.html
3 DJB
http://cr.yp.to
4 DJB
http://people.oven.com/bet/lwd/lwd.html
5 djb-dns
http://cr.yp.to/djbdns.html
6 Článek o djb-dns
http://www.root.cz/clanek.phtml?id=628
7 holelogd
ftp://ftp.obtuse.com/pub/utils
8 RPM balíček bindu v chroot prostředí
ftp://ftp.linux.cz/pub/linux/people/jan_kasprzak/bind-chroot/
9 Securing DNS (Linux Version)
http://www.psionic.com/papers/dns/dns-linux
10 Securing Your Name Servers
http://securityportal.com/direct.cgi?/closet/closet19991124.html
11 Hardening the BIND DNS Server
http://securityportal.com/cover/coverstory20001002.html

Monitorování dostupnosti služeb — MON

David Häring, 1. dubna 2001

Existuje řada nástrojů určených k nepřetržitému monitorování dostupnosti důležitých (nejen) síťových služeb. Jedním z takových nástrojů je i utilita **mon** (1), na kterou se dnes blíže podíváme.

Mon sestává z plánovacího procesu (serveru), který podle konfigurace v určitých intervalech spouští testovací akce. V závislosti na výsledku testů pak v případě potřeby provádí další předem definované akce (např. upozornění prostřednictvím e-mailu) a zapisuje výpadky služeb do logu. Vlastní testy, stejně jako i akce prováděné při výpadku jednotlivých služeb, jsou realizovány samostatnými utilitami (tzv. monitory), které jsou nezávislé na plánovacím procesu monu. Dodejme ještě, že mon je napsán v Perlu a přímo v distribuci monu je řada utilit pro testování běžně používaných služeb. Toto uspořádání je výhodné, protože veškeré pomocné utility lze snadno uzpůsobit „na míru“ místním potřebám či přidat nové bez nutnosti zasahovat do vlastního serveru monu.

Distribuce monu (2) kromě vlastního plánovacího serveru obsahuje utility `moncmd` (utilita pro příkazovou řádku, slouží pro ovládání serveru — server je možno restartovat, zapínat či vypínat monitorování skupin nebo jednotlivých strojů, upravovat konfiguraci za běhu apod.) a `monshow` (slouží k prohlížení stavu monitorovaných služeb; utilita je buď pro použití z příkazové řádky, anebo ji lze použít přes www jako CGI aplikaci). Mezi monitory obsaženými v distribuci monu jsou například monitory služeb SMTP, FTP, TELNET, HTTP, POP-3, IMAP, DNS, LDAP, SQL.

Instalace

Mon pracuje s řadou perlovských modulů, které zpravidla nejsou k dispozici v obvyklých instalacích Perlu. Seznam modulů, které jsou zapotřebí, je uveden v souboru

```
# Definujeme dvě skupiny strojů (servers a nameservers)
hostgroup servers server1.domena.cz server2.domena.cz server3.domena.cz
hostgroup nameservers 11.22.33.44 11.22.33.45

# Ve skupině servers monitorujeme služby ping, http, ssh, mySQL

watch servers
# ping každé dvě minuty, notifikace e-mailem pokud je systém alespoň 2x
# nedostupný v intervalu 10 minut
    service ping
        interval 2m
        monitor fping.monitor
        period wd {Sun-Sat}
            alert mail.alert admin@domena.cz
            upalert mail.alert -S "server is alive" admin@domena.cz
            alertafter 2 10m
            alertevery 2h
            upalertafter 4m
            numalerts 5
# http monitorujeme každé tři minuty, notifikace e-mailem pokud je systém
# alespoň 2x nedostupný v intervalu 15 minut
    service http
        interval 3m
        monitor http.monitor
        depend servers:ping
        period wd {Sun-Sat}
            alert mail.alert admin@domena.cz
            upalert mail.alert -S "http service is back up" admin@domena.cz
            alertafter 2 15m
            alertevery 2h
            numalerts 5
# Službu ssh monitorujeme 1x za hodinu, každý výpadek je hlášen e-mailem.
# Monitor se připojí na port SSH a kontroluje
    service ssh
        interval 1h
        monitor ssh.monitor
        depend servers:ping
        period wd {Sun-Sat}
            alert mail.alert admin@domena.cz
            upalert mail.alert -S "ssh service is back up" admin@domena.cz
            alertevery 2h
            numalerts 5

# Dostupnost SQL databáze monitorujeme jednou za 4 minuty, notifikace e-mailem
# pokud je systém alespoň 2x  nedostupný v intervalu 15 minut. Testování probíhá
# tak, že se monitor připojí k databázi a vypíše seznam tabulek
    service mysql
        interval 4m
        monitor mysql.monitor --username=testuser --password=testpswd --database=dbtest
        depend servers:ping
        period wd {Sun-Sat}
            alert mail.alert admin@domena.cz
            upalert mail.alert -S "SQL server is back up" admin@domena.cz
            alertafter 2 15m
            alertevery 2h
            numalerts 5
```

Výpis č. 4: Ukázka části konfiguračního souboru `mon.cf`

`INSTALL`; k dispozici jsou na kterémkoliv zrcadle CPANu (3) Kromě toho je zapotřebí instalovat také klientskou část monu, která je rovněž dostupná jako modul na CPANu anebo na domovské stránce monu.

Pokud máme všechny požadované moduly instalovány, můžeme přikročit k instalaci serveru. Po rozbalení distribuce v podstatě stačí pouze vytvořit konfigurační soubory `/etc/mon.cf` a `/etc/auth.cf` (můžeme vyjít z příkladů konfigurací přiložených v distribuci) a přidat záznam do `/etc/services` (detaily viz soubor `INSTALL` v distribuci monu).

Na závěr zbývá volitelně přeložit některé monitory, kte-

```
# Ve skupině nameserver monitorujeme dostupnost DNS tak, že každou hodinu stahujeme
# SOA záznam z primárního serveru pro zónu domena.cz (11.22.33.44). Následně je
# porovnáno sériové číslo zóny mezi primárním a sekundárními servery (11.22.33.45)
watch nameserver
    service dns
        interval 1h
        monitor dns.monitor -zone domena.cz -master 11.22.33.44 11.22.33.45
        period wd {Sun-Sat}
            alert mail.alert admin@domena.cz
            upalert mail.alert -S "DNS server is back up" admin@domena.cz
            alertevery 2h
            numalerts 5
```

Výpis č. 4: Ukázka části konfiguračního souboru `mon.cf` (pokračování)

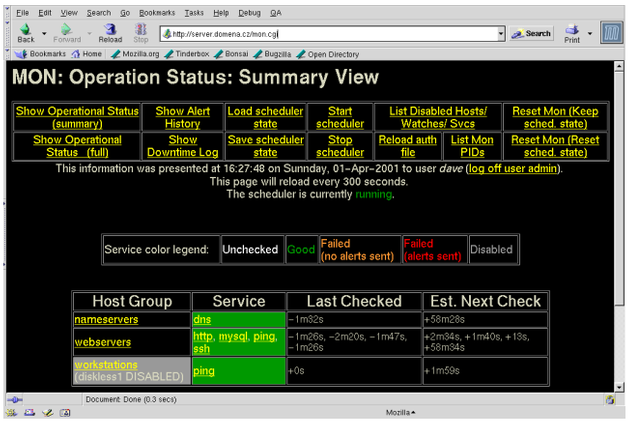
```
     server: localhost
       time: Sun Apr 1 10:56:30 2001
      state: scheduler running
  GROUP           SERVICE      STATUS      LAST       NEXT       ALERTS SUMMARY
R nameservers     ping         -           28 secs    00:01      none
D nameserers      dns          -           00:33 ag   00:26      none
R servers         ping         -           6 secs a   00:01      none
D servers         http         -           00:02 ag   00:01      none
D servers         mysql        -           00:03 ag   58 secs    none
D servers         ssh          -           00:30 ag   00:29      none
```

Výpis č. 5: Ukázka výpisu statusu monu — (příkaz monshow –full)

ré jsou napsány v C (týká se např. monitoru RPC služeb), pokud tyto monitory nehodláme používat, můžeme tento krok přeskočit.

Konfigurace

Konfigurační soubor hledá mon standardně v `/etc/mon.cf`. Kromě obecných nastavení cest k souborům, metod autentizace uživatelů apod. se zde nachází definice služeb, které jsou monitorovány. Syntaxe konfiguračního souboru je jednoduchá, proto pro ilustraci uvedeme pouze okomentovaný příklad části konfiguračního souboru (Ukázka části konfiguračního souboru `mon.cf`).



Autentizace uživatelů

Pokud jde o kontrolu přístupu k monu, mon podporuje tři typy autentizace: přímé použití souboru `/etc/passwd`, hesla uložená ve zvláštním souboru (formát obdobný jako používá web server **apache**) a autentizace pomocí PAM. Mon neumí pracovat s hesly ukládanými v souborech `/etc/shadow`, což ale vzhledem k tomu, že většina distribucí dnes již PAM podporuje nevadí. Seznam akcí, které jsou jednotliví uživatelé oprávněni provádět je uložen odděleně v souboru `auth.cf`.

Mon neumožňuje omezení přístupu podle jmen strojů či IP adres. Kromě používání autentizace uživatelů je z hlediska zabezpečení určitým řešením zajistit pomocí direktiv `serverbind` a `trapbind`, aby se mon navázal pouze na porty rozhraní loopback (127.0.0.1). Flexibilnější a bezpečnější řešení je použití firewallu.

Webové rozhraní

První z možností je použít utilitou monshow, viz Ukázka výpisu statusu monu — (příkaz monshow –full), která je součástí distribuce monu. Monshow může být spouštěn buď z příkazové řádky anebo jako CGI aplikace s tím, že vzhled webového rozhraní lze definovat pomocí tzv. náhledů („views“). Dvěma alternativními nástroji jsou **mon.cgi** (4) a **Minotaure**(5).

Mon.cgi zpřístupňuje veškerou funkčnost monu včetně administrace. Instalace je jednoduchá — v podstatě pouze rozbalíme skript do příslušného adresáře, upravíme a nakonfigurujeme web server. Mon.cgi při komunikaci používá *cookies*, ve kterých je uloženo uživatelovo heslo zašifrované algoritmem TripleDES, takže i v případě, že nepoužijeme https, heslo putuje v nezašifrované podobě mezi klientem a serverem pouze jednou. Na obrázku vidíme mon.cgi v akci.

Závěrem

Mon je velmi dobře konfigurovatelný nástroj, navíc kromě

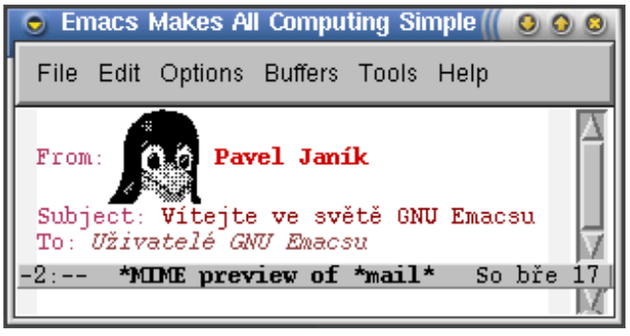
řady monitorů běžně používaných služeb není problém vytvořit si další monitory podle potřeby. Také existuje diskusní fórum uživatelů monu (6) včetně archívu (7), kde lze řešit případné problémy. Hledáte-li vhodný nástroj pro monitorování dostupnosti síťových služeb, mon určitě stojí za povšimnutí. ■

1 Mon
http://www.kernel.org/software/mon/
2 Distribuce MONu
ftp://ftp.kernel.org/pub/software/admin/mon/
3 CPAN archív
http://www.cpan.org/
4 Mon.cgi
ftp://ftp.kernel.org/pub/software/admin/mon/contrib/
5 Minotaure
ftp://ftp.kernel.org/pub/software/admin/mon/contrib/
6 Mon - mailing list
http://www.kernel.org/software/mon/list.html
7 Archív mailing listu
http://acsys.anu.edu.au/~tpot/hypermail/mon/

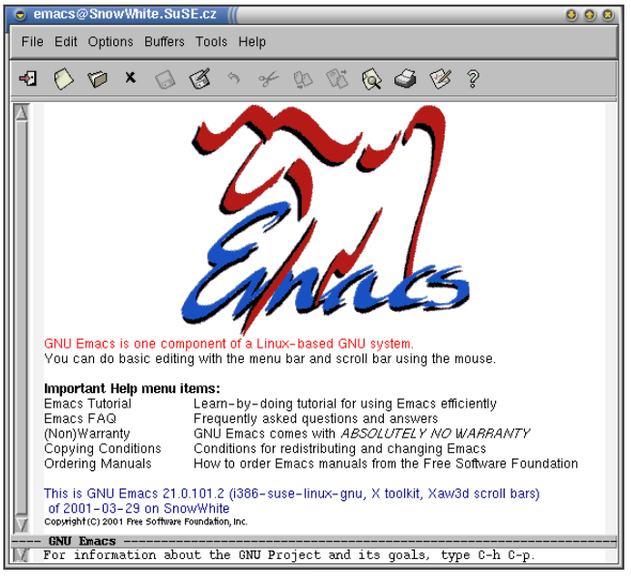
GNU Emacs 21

Pavel Janík, 2. dubna 2001

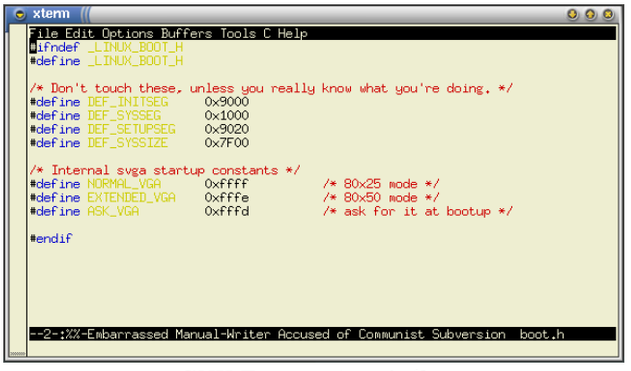
Editor **GNU Emacs** (1) je považován za velmi kvalitní editor. Někteří jej dokonce považují za velmi kvalitní operační systém. V každém případě se ale jedná o vlajkovou loď hnutí free software, která brázdí rozlehlé vody oceánu GNU (2) projektů již hezkou řádku let (Richard Stallman na něm začal pracovat již v září roku 1984) — nejprve pod vedením samotného Richarda Stallmana a nyní pod vedením skromného, ale velmi pracovitého Němce Gerda Möllmanna. Ale tým vývojářů čítá několik desítek lidí rozmístěných po celém světě.



Vývojový mechanismus je na rozdíl od jiných free softwarových projektů poměrně uzavřený, probíhá v CVS, které není veřejně přístupné, a pouze vždy několik málo měsíců před uvolněním nové verze je připraveno několik tzv. pretest verzí, jež se dostanou mezi úzkou skupinku programátorů píšících doplňující módy pro GNU Emacs nebo se jinak podílí na vývoji. Nyní se právě nacházíme v této době, kdy již několik pretest verzí bylo zveřejněno a právě probíhají poslední korektury rozsáhlého manuálu (má přibližně 630 stran). Stávající vývojový model se stal v minulosti terčem velmi silné kritiky, a proto bude s velkou pravděpodobností otevřen ihned po uvolnění verze 21. Já jsem ale skeptik a nemyslím si, že by to vývoj usnadnilo. Současný vývojářský tým je velmi schopný a na konkrétní a úplné popisy chyb s popisem jejich snadné reprodukce odpovídá i do několika minut, a to většinou záplatou, která je oproti poslední verzi v CVS stromu. Nicméně otevřením vývoje dojde k tomu, že nový kód bude testovat více lidí a ti budou posílat více nekonkrétních a neúplných hlášení chyb a to pravděpodobně způsobí u vývojářů mírné znechucení, které bude spíše brzdou rychlosti vývoje. Nicméně uvidíme. Třeba bude všechno jinak.



Poslední uvolněná verze GNU Emacsu má číslo 20.7 a všichni čekají na novinky, které přinese verze 21. Dokonce se již objevují první kousky kódu, které jsou určeny pouze pro verzi 21.1.



GNU Emacs v terminálu

Novinky

Copak bude ve verzi 21 nového? Především podstatně vylepšená podpora práce ve více jazycích. GNU Emacs bude konečně brát ohled na nastavení locale, a proto uživatelé, kteří mají nastavené české (či slovenské) jazykové prostředí např. pomocí proměnné LANG, nebudou mít prakticky žádné potíže. Velmi očekávanou novinkou je také podpora barev na textovém terminálu a v xtermu.

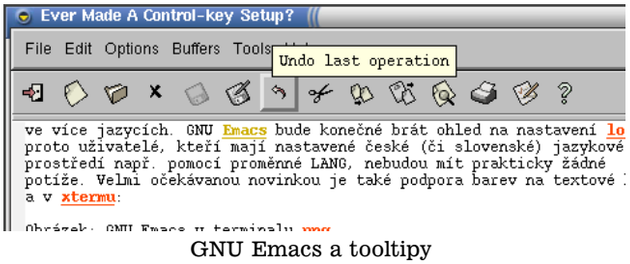
GNU Emacs verze 21 již také umí pracovat s obrázky, a tak Vás nemůže překvapit, že pokud dostanete e-mail s přiloženým obrázkem, tak jej budete moci zobrazit přímo v GNU Emacsu bez volání dalšího externího programu.

Na obrázku vidíte také nové **Gnus** (3), které nazvat jinak

než „message laboratory“ by bylo rouháním :-) Gnus jsou původně program pro čtení news, nyní ale podporují jak Usenet News, tak i klasickou poštu, umožňují sofistikované třídění a další činnosti, které Vás jenom mohou napadnout.

GNU Emacs a obrázky

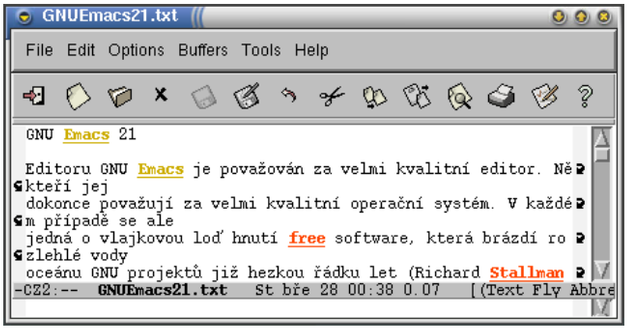
Ale vraťme se zpět k vlastnímu prostředí, které doznalo oproti minulým verzím několika změn. Předně je k dispozici tzv. toolbar, kterou vidíte na předchozím obrázku, s kontextovou nápovědou ve formě tzv. tool-tipů:



GNU Emacs a tooltipy

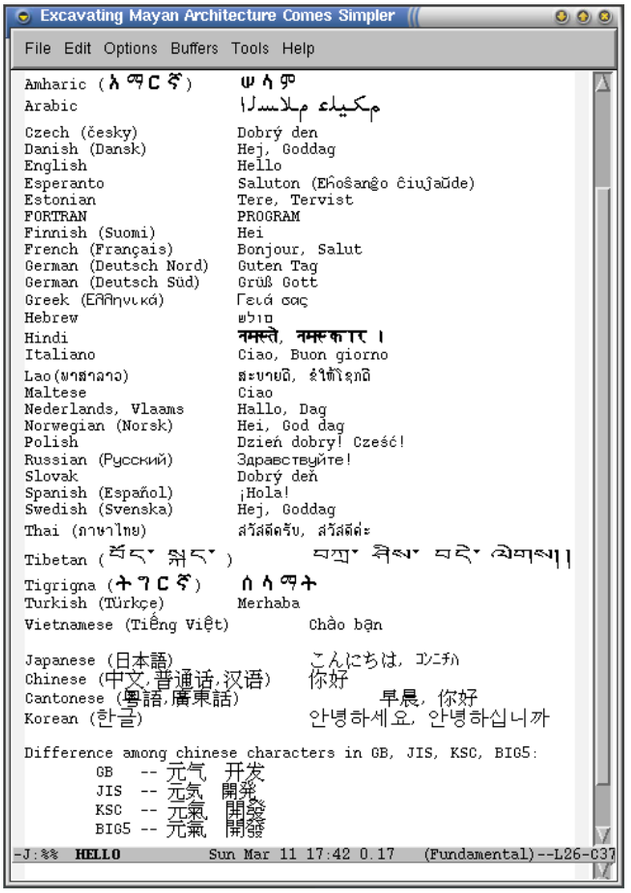
Novinkou je také jednoduchá možnost identifikace řádků delších než okno GNU Emacsu pomocí tzv. *fringe*, tedy sloupečků kolem textu, které budou v případě, že text na řádku je delší než šířka okna, obsahovat malé šipečky.

Novinek je spousta, ale ty hlavní nejsou ve vzhledu GNU Emacsu jako takového, ale v jeho možnostech, které byly výrazně rozšířeny. Přibyla velká spousta nových balíčků (ať již zmiňované Gnus 5.9 s podporou MIME a obrázků, quickurl, todo apod.).



GNU Emacs a dlouhé řádky

Velký krok udělala také česká a slovenská dokumentace. Ale to bychom již mnoho napovídali. Cílem tohoto článku není prozradit vše, co GNU Emacs 21 umí, ale jenom malé představení jeho možností. Ostatně pokud čas dovolí, budeme se s GNU Emacsem na stránkách Linuxových novin setkávat častěji. Třeba si i někdy ukážeme, jak v GNU Emacsu vygenerovat něco, co by se i podobalo následujícímu obrázku. Jste zvědavi?



GNU Emacs a více jazyků v jednom dokumentu

■

1 GNU Emacs
http://www.gnu.org/software/emacs/

2 GNU
http://www.gnu.org
3 Gnus
http://www.gnus.org

Správa projektů pomocí CVS

Miloslav Ponkrác, 29. ledna 2001

Úvod

Pokud to myslíte s prací na nějakém projektu opravdu vážně, po nějaké době začnete pociťovat potřebu nějakého programu, který vám v něm udělá pořádek. Program, který zařídí, abyste si nejnovější verzi nějakého textu, programu, obrázku, či čehokoli jiného nepřepsali starší verzí. Snad každý takto přišel občas o práci, která mu zabrala dlouhý čas. Kromě toho se občas stane, že naopak zjistíte, že byste se potřebovali vrátit ke starší verzi, protože čas ukázal, že je lepší, než novější. Ale co s tím? Schraňovat mnoho starých verzí, kdyby náhodou? A to už vůbec nemluvím o tom, pokud byste na projektu měli dělat dva, či vás budou pracovat desítky lidí na tom samém projektu. Jak potom zabránit zmatku?

Protože tyto problémy měli lidé již dlouho, začaly vznikat různé programy na správu projektů. A tento článek by měl být o léty prověřeném programu, který sám používám, a který si nemohu vynachválit. Jedná se o CVS, což je anglická zkratka **Concurrent Versions System**. Tento systém má za sebou dlouhý vývoj, je k dispozici na většinu platforem, mimo jiné i na Windows, Linux, OS/2 a mnohé další. Tímto systémem byly řízeny mnohé projekty. A to nejlepší nakonec, přes jeho nesporné kvality je zcela zadarmo, můžete si ho stáhnout na stránkách CVS (1).

Snad jenom upozornění, tento dokument je psán tak, že pokud chcete začít pracovat se systémem CVS, anebo ho jenom vyzkoušet, nemusíte ho číst celý. Následující kapitoly vás provedou krok za krokem nejnutnějšími kroky, které musíte udělat. Začínám tím, kde program **cvs** získat, jak ho nainstalovat, jak vytvořit nový projekt, a jak v něm spravovat váš první projekt. Pokud chcete systém CVS používat opravdu jen na nejjednodušší správu projektů, a pracujete na projektu sám, stačí vám toto nezbytné minimum. Pokud potřebujete spravovat projekt ve více lidech, bude nezbytné si přečíst ještě minimálně alespoň kapitolu o práci ve vícečlenném týmu. Zbylé kapitoly vám umožní používat systém CVS s lehkostí a jistotou.

Filozofie práce se systémem CVS

Tato kapitola o filozofii práce se systémem CVS zde původně nebyla. Vzhledem ke zkušenostem s lidmi, kteří začali používat CVS podle tohoto dokumentu jsem později zjistil, že patří mezi nejdůležitější. Proto vás prosím, přečtěte si tuto kapitolu důkladně. Velice vám usnadní pochopení dalšího textu.

První je potřeba pochopit, že systém CVS slouží jako systém, který dohlíží na skupinu souborů nazývaných *projekt*. Systém CVS je schopen udržet pořádek v této skupině souborů a nejenom to. Umožní vám, abyste se třeba kdykoli vrátili ke starší verzi. Není třeba problém chtít po CVS, aby mi nahrál do adresáře stav projektu třeba 1. ledna 1999 o půlnoci. Umožní vám, aby do skupiny souborů (projektu) dělalo zásahy a změny více lidí. Je možné například vyvíjet souběžně více verzí, což je často potřeba. Představte si třeba, že vyvíjíte informační systém, který jste v první verzi dodali zákazníkovi. Mezitím začnete vyvíjet druhou verzi informačního systému. Poté zákazník objeví několik chyb v první verzi, a požaduje opravu. Je jasné, že mu nedáte nestabilní druhou verzi, která se teprve vyvíjí (tedy jste-li profesionálové), ale uděláte opravy nad první verzí. A nyní tu máte souběžně dvě větve projektu. Jedna vede k vývoji stabilní odladěné první verze, a druhá k vývoji zatím nedokončené druhé verze. A systém CVS toho dokáže daleko více.

Na druhé straně je potřeba pochopit, že systém CVS je především systém správy různých verzí souborů. Je zbytečné od něj chtít například přeložit hotový program, na to existují jiné nástroje. Ale určitě vám zaručí, že v programu vám nevzniknou chyby jenom proto, že někdo přeložil starší modul. Systém CVS také neprovádí testování programu, ani nenahrazuje vedoucího projektu.

V systému CVS se pracuje na několika úrovních. Přesněji řečeno, práce se systémem CVS je myšleně rozdělena do několika celkem oddělených kategorií. V zásadě jde o vytvoření nového projektu, administrace projektu, a potom akce, které umožňují rutinní práce. Rutinní prací myslím získání aktuální verze (případně jiné) verze projektu, a poté nahrání nové verze do systému CVS.

Kde získat systém CVS

Při používání systému CVS si můžete vybrat, zda budete s tímto systémem komunikovat pomocí příkazové řádky, a nebo pomocí grafického rozhraní. Je také možné obojí kombinovat.

V době, kdy píši tento text, se zdá, že CVS je spravováno na stránkách cvshome.org (2). Soubory, které jsou potřeba pro správu CVS pomocí příkazové řádky, lze stáhnout z FTP serveru (3). Zde si vyhledejte svůj operační systém (nejčastěji asi Windows, nebo Linux) a stáhněte si poslední stabilní verzi. Konkrétně poslední binární verze pro Windows ke stažení je verze 1.10, kterou lze stáhnout jako soubor `cvs-1.10-win.zip`. Tento soubor obsahuje vše, co budete potřebovat. Pokud používáte Linux, či jiný systém, najděte si příslušný podadresář, a řiďte se zvyklostmi vašeho operačního systému.

Pokud chcete spravovat CVS pomocí grafického prostředí, potom existuje vícero systémů, jejichž popis není součástí tohoto manuálu (alespoň s tím zatím nepočítám). Tuším, že grafické prostředí v pozadí volá CVS s příkazovou řádkou.

Vyzkoušel jsem WinCVS pro Windows, který je velice pěkně udělaný. Pokud chcete WinCVS používat, jeho součástí je i balík pro práci s příkazovou řádkou, i když jej nemusíte používat. WinCVS všude upozorňuje, že není zaručena funkčnost s Windows 95, na této verzi Windows je potřeba instalovat nějaké novější DLL. Moje krátkodobé zkušenosti s WinCVS říkají, že pro zvládnutí základní práce s CVS potřebujete prakticky stejné znalosti, jako pro práci s příkazovou řádkou. Nečekejte proto, že sednete ke grafické nadstavbě a budete okamžitě spravovat projekty, pokud nemáte s CVS žádné zkušenosti. Síla grafické nadstavby se projeví spíše v pokročilejších funkcích, složitější akce provedete rychleji a jednodušeji. A hlavně je to otázka, zda dáváte přednost klikání, nebo příkazové řádce. Pokud vám vyhovuje grafické prostředí více, tak jděte do toho.

V dalším textu budu předpokládat, že budete ovládat systém CVS pomocí příkazové řádky. Jednak je toto ovládání jednotné, a přitom nijak složité a umožňuje vám přitom plnou kontrolu nad systémem. Ovládání pomocí grafické nadstavby se liší podle platformy, na které budete systém používat, a také podle použité grafické nadstavby.

Instalace systému CVS

Jak tedy začít? Stáhněte si tedy nejdříve CVS, viz předchozí kapitola. Získaný soubor umístěte na disk. Pro Windows neprobíhá žádná instalace, pouze soubory ze zazipovaného archivu rozbalíte do vámi zvoleného adresáře kamkoli na disk. Pro ukázku budu v dalším textu předpokládat, že jste umístili cvs do adresáře `c:\cvs` (v případě Windows), případně `/usr/bin` (v případě Unixu).

Nyní je potřeba zvolit adresář, ve kterém si bude CVS systém uchovávat informace o projektech. Je lépe zvolit jiný adresář, než ve kterém je umístěn vlastní program cvs. Volba umístění adresáře závisí na tom, jakým způsobem budete chtít projekty řídit. Pokud chcete řídit projekty vlastní, asi nejlepší je zvolit adresář na vašem lokálním disku. Pokud chcete řídit vývoj projektů ve Vaší firmě, je dobré umístit jej na server Vaší lokální sítě. A pokud chcete vyvíjet například společný projekt po Internetu, je vhodné jej umístit na FTP server. Důležité je, aby všichni, kdo se účastní na projektu, mohli z tohoto adresáře číst i zapisovat. Jsou samozřejmě možné i jiné varianty, například adresář s informacemi o projektech může mít na svém lokálním disku vedoucí projektu, a lidé, kteří se účastní projektu si čas od času přijdou nahrát do systému CVS svoji práci a odnést si změny, které provedli druzí. Je možnost zvolit mnoho jiných variant. Je potřebné se také zamýšlet nad otázkou bezpečnosti. Pokud například nechcete povolit přístup k projektu nepovolaným, je potřeba jej neumísťovat na veřejném FTP serveru, v lokální síti by do adresáře s informacemi o projektu neměli mít přístup nepovolaní, apod.

Pro počáteční vyzkoušení systému CVS je asi nejlepší umístit adresář s informacemi na lokální disk. Zvolíme tedy adresář pro data. Pro Unix jej v rámci ukázky zvolím asi jako `$HOME/cvsroot`, pro Windows třeba jako `c:\cvsroot`. Pro instalaci je potřeba udělat ještě dvě akce, a to přidat program cvs do cesty pomocí proměnné PATH, abychom mohli program cvs spouštět z jakéhokoli adresáře. A nastavit proměnnou CVSROOT, ve které bude zapsána plná cesta k adresáři pro data systému CVS.

Nejlepší cesta, jak nastavit obě proměnné, je v případě Windows zapsat je do souboru `AUTOEXEC.BAT`, v případě Unixu do `.profile`. Úpravy v `AUTOEXEC.BAT` (pro Windows): Na konec souboru `AUTOEXEC.BAT` přidat následující dva řádky. Poté uložit a restartovat.

```
set CVSROOT=:local:c:\cvsroot
set PATH=c:\cvs;%PATH%
```

Úpravy pro Unix ponechám na váženém čtenáři.

V této chvíli je tedy potřeba vytvořit adresář pro data programu. Jde jenom o to, že adresář `c:\cvsroot` (pro Windows), případně `$HOME/cvsroot` (v případě Unixu) musí existovat. Pokud neexistuje, vytvoříme ho, ať už pomocí příkazu `mkdir` (pracuje jak ve Windows, tak v Unixu), a nebo jinak. Máme tedy vytvořen prázdný adresář pro data projektu.

Teď je nutné inicializovat adresář pro data. To slouží k tomu, aby si systém CVS vytvořil potřebné soubory pro správu projektů. Provádí se to příkazem `cvs init`. V této chvíli máme systém CVS připravený k používání.

Založení prvního projektu v systému CVS

V této chvíli máme systém CVS připravený ke správě libovolného množství projektů. Dále je potřeba do něj jednotlivé projekty „nasázet“. Systém CVS můžete použít ke správě například tvorby vašeho Webu, k řízení programátorského projektu, či jiného projektu. Záleží na tom, co potřebujete. Předem je ale potřeba říci, že se jedná o systém ke správě verzí. Dokáže tedy velmi dobře udržovat pořádek v různých verzích, i když na projektu pracuje mnoho lidí, ale rozhodně není určen pro překládání programů do binární podoby apod.

Předpokládejme tedy, že budeme vytvářet webovskou stránku. Tento projekt si nějak pojmenujeme, například „prvniprojekt“. Vytvoříme si někde adresář, který si může jmenovat naprosto jakkoliv. Jméno adresáře dokonce nemusí mít vůbec nic společného s názvem projektu, jenom do něj překopírujeme vše, co již na tomto projektu máme hotovo. Pokud začínáme, a nemáme zatím nic, nechme adresář prázdný. Důležité je, aby v tomto adresáři byly pouze soubory, které se týkají našeho projektu. Soubory mohou být rozdělené do podadresářů. Pro začátek doporučuji, aby se jednalo pouze o textové soubory (tedy texty, HTML stránky, zdrojové kódy programů, skripty apod.). Ostatní soubory zatím do adresáře neumisťujte, přidáme je později, potřebují trochu odlišný způsob. Zadání projektu prvniprojekt do systému CVS provedeme ve dvou krocích:

1. Přesuneme se do adresáře, kde je vše co patří k projektu. Jak již bylo napsáno výše, pokud začínáme s projektem, může se jednat i o prázdný adresář.
2. Z tohoto adresáře spustíme příkaz:

```
cvs import -m "zalozeni projektu"\
 prvniprojekt poc-tym poc-verze
```

Vysvětlím, co znamenají podrobněji části výše uvedeného příkazu. Slovo `import` znamená příkaz, který systém CVS vyzývá k založení nového projektu. Zároveň mu říká, aby do tohoto projektu zavedl všechny soubory, které objeví v aktuálním adresáři (zjistíte příkazem cd ve Windows, případně pwd v Unixu). Do projektu se také přidají všechny podadresáře, a soubory v těchto podadresářích. Volba `-m` říká systému CVS, že chceme připsat poznámku, která je uvedena v uvozovkách za touto volbou. Musím podotknout, že poznámku v tomto případě systém CVS vyžaduje, a pokud mu ji nezadáte volbou *-m* v příkazovém řádku, systém CVS spustí editor, kde mu ji můžete zadat. Slovo `prvni-projekt` je název projektu. Další dvě slova značí jméno týmu a jméno počáteční verze, tam můžete zadat cokoli, třeba zrovna „poc-tym“, nebo „poc-verze“.

Pokud jste se dostali až sem, a spustili příkaz, obvykle vám systém CVS nejdříve ukáže průběh načítání souborů do projektu, a potom napíše něco v tom smyslu, že „No conflicts“ a ještě něco dalšího. Založili jste úspěšně projekt do systému CVS.

Teď bude dobré smazat adresář, ze kterého jsme spustili CVS, a zkopírovali soubory patřící k projektu. Protože zatím se systémem začínáte, nezapomeňte si soubory zálohovat! Systém CVS je prověřený a celkem spolehlivý, ale je potřeba s ním umět pracovat.

Správa prvního projektu v systému CVS

Máme tedy první projekt uložený v systému CVS. Kdykoli budeme chtít na projektu pracovat, přepneme se do adresáře, ve kterém chceme pracovat. V tomto adresáři spustíme příkaz:

```
cvs checkout prvniprojekt
```

Tento příkaz udělal to, že vytvořil v pracovním adresáři podadresář `prvniprojekt`, a do něj nahrál aktuální verzi všech souborů. Kromě toho si do každého adresáře přidal podadresář CVS se svými údaji. Nyní můžete v souborech měnit, co se vám líbí. Opravovat chyby, dopisovat, zkrátka pracovat na projektu. Po skončení práce je potřeba vaše změny nahrát zpět do systému CVS, a to příkazem:

```
cvs commit -m "nejaka poznamka"
```

Příkaz `commit` zaznamenává změny na projektu zpět do systému CVS. Teď se možná ptáte, co je na tom tak úžasné. Třeba to, že na projektu může pracovat více lidí najednou, a systém CVS si dokáže poradit i v případě, že více lidí změní stejný soubor. A nebo vám dokáže vytáhnout stav projektu, jaký byl v libovolné chvíli. Chcete získat stav souborů přesně tak, jako byl třeba 2. prosince? Není to problém. A to je jenom velice malá přehlídka jeho možností. Nikdy žádnou změnu neztratí, a přitom zbytečně neplýtvá diskovým prostorem. Ukládá si totiž pouze změny od předchozí verze, takže jeho data o projektu zabírají poměrně málo místa na disku.

Nutno říci, že příkaz `commit` je daleko flexibilnější, než bylo uvedeno. Můžete například uložit změny pouze jednoho souboru:

```
cvs commit -m "nejaka poznamka" jmeno_souboru
```

Při používání příkazu `commit` zjistíte, že systém CVS kontroluje pouze soubory, které tam už byly při zakládání projektu. V pracovním adresáři si tedy můžete vytvořit spousty souborů, ale systém CVS si všimne jenom těch, které tam byly od počátku. Pokud potřebujete přidat nový soubor do systému CVS, použijte příkaz spuštěný z pracovního adresáře:

```
cvs add -m "nejaka poznamka" jmeno_souboru
```

Systém CVS pracuje se skupinou souborů nazývaných projektem, jak už bylo řečeno výše. Každý soubor, který má být součástí projektu je nutné zaregistrovat, aby CVS věděl, že má hlídat změny i tohoto souboru. To se provádí dvěma způsoby. Prvním je příkaz `import`, který provede vytvoření nového projektu v systému CVS, a zároveň zaregistruje všechny soubory v aktuálním adresáři i jeho podadresářích. Příkaz `import` je určen ke kompletnímu vytvoření projektu, to znamená, že navíc všechny soubory nahraje do projektu.

Pokud již máme projekt v systému CVS vytvořený, používá se druhý způsob, a to příkaz `add`. Tento příkaz zařídí, že systém CVS bere na vědomí, že součástí projektu se stal další soubor, a nic víc! To znamená, že aktuální verze nového souboru není nahrána do systému CVS! To se provede až dalším příkazem, který slouží k nahrávání změněných souborů do systému CVS, a tím je `commit`. Tedy ještě jednou: po použití příkazu `add` si systém CVS pouze poznamená, že od této chvíle bude součástí projektu i nově zadaný soubor. Tento soubor si ale nenahraje do systému CVS, protože by v tom byl zmatek. Jediný příkaz, který umožňuje nahrávat změny souborů do systému CVS, je příkaz `commit`. Proto vám příkaz `add` vypíše (v angličtině) zprávu, že si poznamenal, že soubor patří do projektu, ale počká si na příkaz `commit`, který tento soubor nahraje do systému. A poté již bude automaticky nahrávat i změny v novém souboru od všech lidí, kteří pracují na projektu.

Teď již tedy víte, že každý nový soubor je potřeba do projektu zaregistrovat. Dát prostě systému na vědomí, aby nový soubor také považoval za součást projektu. Při registrování je potřeba upozornit na dvě důležité skutečnosti. Za prvé, pokud vytvoříte nový podadresář v projektu, je potřeba ho také zaregistrovat pomocí příkazu `add`. Provádí se to úplně stejně, jako kdyby to byl soubor. A až potom je možné zaregistrovat soubory v tomto podadresáři. Pokud tedy v pracovním adresáři založíme nový podadresář, a v něm nový soubor, je potřeba nejdříve přidat do systému CVS podadresář, a až poté nový soubor. Podadresář se přidává úplně stejně jako soubor, jenom místo jména souboru uvedeme jméno adresáře.

Druhou důležitou skutečností je to, že musíte dát systému CVS na vědomí, co je čistý textový soubor, a co je binární soubor. Binárním souborem se rozumí například obrázky, dokumenty Office, i když je to z Wordu, apod. Pokud se jedná o čistý text, nic zvláštního se neděje, pokud budete pracovat s binárními soubory, je nutné při registraci tohoto souboru dát systému CVS na vědomí, že to není čistý text! V takovém případě je při registrování pomocí příkazu `add` nutné použít volbu `-kb`. V textových souborech totiž provádí CVS určité optimalizace. Příkaz na přidání binárních souborů vypadá tedy takto:

```
cvs add -kb -m "nejaka poznamka" jmeno_souboru
```

Pokud přidáte binární soubor, například obrázek, aniž byste systému CVS sdělili, že se nejedná o čistý text, systém CVS takový soubor zdeformuje. Je tedy potřeba si na toto dávat pozor. Pokud čistě náhodou na to zapomenete, a přidáte binární soubor bez volby `-kb`, tak se to dá opravit následujícím způsobem:

- Zazálohujete si soubor z pracovního adresáře, který chcete označit jako binární. Je to proto, že v průběhu této akce se soubor zdeformuje a bude potřeba ho obnovit. Pokud je i v pracovním adresáři soubor zdeformovaný (to se stane, pokud jste předtím použili příkaz `update`), potom je potřeba někde mít čistou, nezdeformovanou verzi souboru, kterou použijete v pátém kroku pro obnovení souboru.
- Provedete příkaz: `cvs admin -kb jmeno_souboru`. Tento příkaz říká, že soubor `jmeno_souboru` bude od této chvíle označen v databázi systému CVS (tzv. repository) jako binární, a CVS s ním takto bude zacházet.
- Provedete příkaz: `cvs update jmeno_souboru`. Příkaz update slouží k tomu, aby si CVS upravil ve vašem pracovním adresáři své vnitřní záznamy o souboru. Ve druhém kroku jste totiž označili soubor jako binární v repository, ale nikoli ve vašem pracovním adresáři. Tento třetí krok proto přenese označení na binární soubor i do vašeho pracovního adresáře. Pokud tento třetí krok neprovedete, tak Vás bude CVS upozorňovat na chybu.
- Třetí krok jako vedlejší efekt nahrál do pracovního adresáře zdeformovaný soubor. Proto je potřeba jako další krok nahrát ze zálohy dobrou, nezdeformovanou verzi souboru.

- Jako poslední krok nahrajeme do systému CVS (do jeho databáze, čili repository) správnou verzi z pracovního adresáře. To se provádí již známým příkazem `commit`: `cvs commit -m "oprava na binární verzi" jmeno_souboru`.

Někdy se naopak stane, že potřebujeme nějaký soubor vymazat z projektu. Potom se nám hodí příkaz `remove`:

```
cvs remove jmeno_souboru
```

A opět je potřeba zdůraznit, že příkaz `remove` nedělá nic jiného, než odregistrovává soubor z projektu. Říká tedy, že soubor přestal být součástí projektu, a systém CVS nebude již dále zaznamenávat změny v tomto souboru. Samozřejmě, že pokud budete chtít vytáhnout ze systému CVS verzi z doby, dokud byl tento soubor součástí projektu, tak ho tam systém CVS přidá, ale ve verzích nahraných do CVS po použití příkazu `remove` ho přestávají zajímat změny na odregistrovaném souboru.

Příkaz `remove` nikde nic nemaže, a v pracovním adresáři vám vše nechá v původním stavu. Ale pro úspěšný průběh příkazu `remove` bude potřeba, aby odstraňovaný soubor v pracovním adresáři nebyl. Pokud budete chtít odstranit soubor, který je přítomný v pracovním adresáři, CVS nic neprovede, a pouze vám vypíše hlášení v anglickém jazyce. Proto je klasický postup pro mazání nejdříve smazat soubor z pracovního adresáře, a poté ho odstranit příkazem `remove`. Aby se nám nezauzlovaly prsty, pokud budeme rušit více souborů, nabízí CVS parametr `-f`. Ten provede navíc smazání souboru z pracovního adresáře, a udělá tedy oba kroky naráz:

```
cvs remove -f jmeno_souboru
```

Toto je v zásadě to nejdůležitější, co potřebujete ke správě projektu pomocí systému CVS, zejména pokud ho spravujete sami. Další kapitoly se zabývají některými dalšími finesami systému CVS. Toto vám stačí, abyste si řídili svůj projekt sami doma. Další čtení vám umožňuje ze systému CVS získat více, nebo pracovat ve vícečlenném týmu, ale v zásadě pokud jste dočetli sem, jste schopni si řídit svůj vlastní projekt v jednočlenném týmu sami doma.

Výpisy změn v souborech — logy

V této kapitole si povíme něco o zjišťování změn v souborech uložených do systému CVS. Systém CVS ve svých manuálech nazývá prostor, ve kterém ukládá data, termínem repository.

Pokud jsme již soubor několikrát změnili, někdy bývá užitečné zjistit, jaké změny jsme provedli. Pokud jste již práci se systémem CVS zkusili, zjistili jste, že vás CVS nutí každou změnu okomentovat. Většina příkazů má možnost zadaní volby -m, za kterou následuje v uvozovkách poznámka. Pokud poznámku nezadáte, CVS spustí editor, abyste ji dopsali. Mírně ironicky by se dalo napsat, že vás „nevtíravým“ způsobem nutí, abyste ke každé změně dopsali komentář, co jste to vlastně udělali.

Tyto komentáře mají svůj smysl, jsou totiž přístupné všem. Kromě toho CVS při každé změně zaznamenává datum a čas změny, kdo změnu provedl, zvýší automaticky číslo verze, a další údaje. Kromě toho umožňuje tyto údaje doplnit i do textu v souborech, které jsou součástí projektu.

Začneme tedy něčím jednoduchým. Jak zjistit celou historii změn nějakého souboru? Stačí na to následující příkaz:

```
cvs log jmeno_souboru
```

Příkaz `log` vypisuje všechny možné informace o souboru, který si CVS udržuje. Můžete si tam přečíst i zadané komentáře, kdo změny provedl (tedy ne že by tam bylo přímo jméno, ale objeví se tam přihlašovací jméno toho, kdo spustil `cvs commit`). Samozřejmostí jsou data, čas, apod. Snad jenom poznámku, příkaz `log` je možné spustit i bez udání jména souboru, potom vypíše informace o všech souborech, což je pěkně rozsáhlé.

Vzhledem k rozsahu logovacího výpisu i pro jeden soubor se většinou výstup ještě nějak upravuje. Buď se za příkaz připojí filtr `more`, který zajistí, že se text zastaví po každé obrazovce, a neodjede:

```
cvs log jmeno_souboru | more
```

Také je možné logovací výpis uložit do souboru:

```
cvs log jmeno_souboru > \
jmeno_souboru_pro_logovaci_vypis
```

Logovací výpisy můžeme řídit různými volbami. Jak si můžeme snadno ověřit, jsou logovací výpisy velmi rozsáhlé. Mnohdy potřebujeme pouze seznam souborů uložených v repository (tedy zaregistrovaných souborů a adresářů). Toho dosáhneme takto:

```
cvs log -R
```

Další volbou je zobrazit pouze změny souborů, které provedl určitý člověk. Předem je potřeba říci, že CVS ukládá s každou změnou provedenou pomocí příkazu `commit` automaticky i informaci, kdo změnu nahrál. Není to zde ale uloženo jako jméno a příjmení, ale CVS si zjistí, kdo je k počítači momentálně přihlášený, a zapíše si toto přihlašovací jméno. Pokud je potřeba zjistit, které všechny změny má na svědomí uživatel s přihlašovacím jménem sandokan, použijte:

```
cvs log -wsandokan jmeno_souboru
```

Jak je vidět z příkladu, použijete volbu `-w`, za kterou bez mezery následuje přímo přihlašovací jméno uživatele. Vynecháte-li jméno souboru, zobrazí se všechny změny v celém projektu.

Pokud vám vadí, že se vypisují i soubory v podadresářích, pomocí volby `-l` (malé písmeno L, nikoli číslice 1) zajistíte, že se budou vypisovat pouze soubory v aktuálním adresáři.

Velmi často Vás také zajímají změnu provedené v určitém čase, třeba za minulý měsíc (aby se vědělo, kdo si zaslouží prémie :-)), před dvěma týdny apod. K tomu je určena volba `-d`, která má velice bohaté možnosti. V následujících příkladech jsou některé z nich uvedeny:

- `cvs log -d "1 week ago"`
  vypíše změny provedené dříve, než minulý týden
- `cvs log -d "yesterday"`
  vypíše změny provedené dříve, než včera
- `cvs log -d "last month"`
  vypíše změny provedené dříve, než za poslední měsíc
- `cvs log -d "2 hours ago"`
  vypíše změny provedené dříve, než před dvěma hodinami
- `cvs log -d "1999/12/01"`
  vypíše změny provedené do 1. prosince 1999
- `cvs log -d "1999/12/01 12:43"`
  vypíše změny provedené do 1. prosince 1999 ve 12:43

• `cvs log -d "> 1 week ago"`
vypíše změny provedené později, než před týdnem

• `cvs log -d ">= 2 hours ago"`
vypíše změny provedené před dvěma hodinami, a nebo později

• `cvs log -d "1999/11/01 <= 1999/11/30"`
vypíše změny provedené od 1. listopadu do 30. listopadu 1999

Výše uvedené informace lze kombinovat, takže například, chcete-li vědět, co všechno změnil uživatel s přihlašovacím jménem abcdefgh za poslední týden v současném adresáři, použijete:

```
cvs log -wabcdefgh -d ">= last week" -l
```

Automatické přidávání informací o verzi do souboru, nahrazování klíčových slov

Pokud vyvíjíme nějaký projekt, stane se někdy, že se ptáme, a jakou verzi souboru máš? To se dá zjistit pomocí data a času, ale není to ono. Jak tedy zajistit, aby v každém souboru byly napsány potřebné informace, třeba zrovna číslo verze? Nebo další údaje?

Systém CVS toto umožňuje pomocí klíčových slov. Pokud kamkoli do nějakého textového souboru, který je součástí projektu napíšete třeba:

```
$Date$
```

a nahrajete změny příkazem `commit`, zjistíte, že systém CVS vám do souboru přidal celé datum a čas poslední změny souboru:

```
$Date: 2000/01/09 17:58:58 $
```

Samozřejmě takových údajů je celá řada, a jsou vyjmenovány a popsány v následujícím seznamu:

- `$Author$` — přihlašovací jméno uživatele. CVS to nahradí zhruba takto: `$Author: ponkrac $`
- `$Date$` — datum a čas poslední změny souboru. Datum a čas je vypisován ve světovém čase, tedy v čase na nultém poledníku. To je také oficiálně používaný čas na Internetu, který je zvolen proto, aby nebyly problémy s posuny času na různých místech zeměkoule. CVS to nahradí zhruba takto: `$Date: 2000/01/09 17:58:58 $`
- `$Header$` — standardní hlavička, která obsahuje plnou cestu k souboru v repository, číslo verze, datum a čas na nultém poledníku, stav souboru, a pokud je soubor uzamčen, tak kdo ho uzamkl. CVS to nahradí za: `$Header: d:\cvsroot/ponny_html/cvs_manual.html,v 1.30 1999/12/31 12:58:15 ponkrac Exp $`
- `$Id$` — to samé, co `$Header$`, ale namísto plné cesty k souboru se vypisuje pouze jméno souboru. CVS to nahradí za: `$Id: cvs_manual.html,v 1.30 1999/12/31 12:58:15 ponkrac Exp $`
- `$Name$` — jméno značky (neboli tagu), pokud existuje. O značkách se můžete dočíst v jedné z dalších kapitol. CVS to nahradí za: `$Name: jmeno_znacky $`
- `$Locker$` — přihlašovací jméno toho, kdo uzamkl verzi, nebo nic, pokud soubory uzamčeny nejsou. Nutno říci, že uzamykání se málokdy používá. CVS to nahradí za: `$Locker: sandokan $`
- `$Log$` — vloží prakticky ty samé informace, které můžete získat příkazem `log`. CVS nahrazuje slovo rozsáhlým výpisem o poslední změně v souboru, takže následující příklad nahrazení je na více řádek: `$Log: cvs_manual.html,v $ Revision 2.11 2000/01/09 17:58:58 ponkrac prosel kontrolou spravnosti syntaxe Revision 1.30 1999/12/31 12:58:15 ponkrac podrobnejsi rozepsani klicovych slov`
- `$RCSfile$` — jméno souboru v repository. CVS to nahradí za: `$RCSfile: cvs_manual.html,v $`
- `$Revision$` — číslo verze. CVS to nahradí za: `$Revision: 2.11 $`
- `$Source$` — jméno souboru v repository s plnou cestou. CVS to nahradí za: `$Source: d:\cvsroot/ponny_html/cvs_manual.html,v $`
- `$State$` — stav souboru ve zkratce, možné stavy souboru jsou `Exp` (experimentální verze), `Stab` (stabilní verze) a `Rel` (verze určená k distribuci). CVS to nahradí za: `$State: Exp $`

Asi nejčastěji se používá klíčové slovo `$Id$`. Rozepíšeme si jeho výstup podrobněji. Pokud do textu souboru dopíšeme toto slovo, učiníme tak do komentářů. Například v HTML souboru bude vypadat asi takto:

```
<HTML>
<!-- $Id$ -->
a tak dále
```

Pokud provedeme příkaz `commit`, objeví se rozvinutý text namísto `$Id$`:

```
<HTML>
<!-- $Id: cvs_manual.html,v 2.11\
 2000/01/09 17:58:58 ponkrac Exp $ -->
a tak dále
```

Zde vidíte všechny podstatné údaje o souboru přímo v něm. Vidíte, že soubor se jmenuje `cvs_manual`. V repository si CVS systém za jméno souboru ukládá čárku a písmeno v. Takže v repository je tento soubor uložen jako `cvs_manual.html,v`. Dále pokračuje číslo verze, v našem případě 1.1. Poté následuje datum a čas na nultém poledníku, přihlašovací jméno toho, kdo jej uložil, a stav verze, kde `Exp` znamená experimentální verzi, `Stab` stabilní verzi a `Rel` verzi určenou k distribuci. Tak jsou všechny informace po ruce přímo v souboru. Podíváte-li se do mnohých kódů například pro Linux, mnohdy tam klíčové slovo *Id* najdete.

Je jasné, že pokud použijete automatické nahrazování klíčových slov v nějakém souboru, či ve všech, je potřeba klíčová slova dávat do poznámek, aby nenarušily funkci souboru. Například v jazyce C můžete napsat `/* $Id$ */`, apod. Dále je potřeba dávat pozor, aby se v souboru náhodou nevyskytly řetězce, které CVS pozná jako klíčové slovo (pokud to není váš záměr). Například já v tomto manuálu nemůžu rovnou napsat Id, protože by po tomto řetězci CVS okamžitě „skočil“, a nahradil jej. Musím využít toho, že znak $ se může v HTML zapsat jako $, což už CVS nechá na pokoji. Pokud bych chtěl to samé zapsat v jazyce C, aniž by mi to CVS nahradil, musel bych psát třeba `char mytxt[] = "\x24Id\x24";`, aby mi ho CVS ne-

chal na pokoji. Dlužno ovšem napsat, že takové situace se v praxi vyskytují velice, velice zřídka, takže problémy vám to nejspíše dělat nebude. V krajním případě, pokud nelze jinak, je možné nahrazování klíčových slov zakázat. Jak to lze provést, se dozvíte ještě v této kapitole.

Občas se vám nehodí, aby systém CVS automaticky nahrazoval klíčová slova, a tak máte možnost tento proces řídit. Standardně CVS nahrazuje klíčová slova ve všech souborech v projektu. CVS umožňuje ke každému souboru sdělit, jaký režim nahrazování klíčových slov použijete. Tento režim je možné určit pomocí příkazů `add` při registraci souboru, a nebo pomocí příkazu `admin` ho kdykoli později změnit:

• `cvs add -krežim jmeno_souboru`
nastaví režim nahrazování klíčových slov při registraci souboru

• `cvs admin -krežim jmeno_souboru`
nastaví režim nahrazování klíčových slov kdykoli později

U volby `-krežim` se mohou použít tyto možnosti:

- `-kkv` — standardní nahrazování klíčových slov, ale nepřidává přihlašovací jméno toho, kdo zamkl soubor. Příklad: `$Id: cvs_manual.html,v 1.30 1999/12/31 12:58:15 ponkrac Exp $`
- `-kkvl` — standardní nahrazování klíčových slov, ale navíc přidává přihlašovací jméno toho, kdo zamkl soubor, pokud je zamknutý. Příklad: `$Id: cvs_manual.html,v 1.30 1999/12/31 12:58:15 ponkrac Exp sandokan $`
- `-kk` — tato volba je antinahrazování, a pracuje naopak. Smaže vše co bylo nahrazeno, a nechá jenom prázdná klíčová slova. Pokud třeba v souboru máte `$Author: ponkrac $`, potom po změně tam zůstane jenom `$Author$`. Příklad: `$Id$`
- `-kv` — nahrazuje tak, že smaže klíčové slovo, a nahradí ho příslušným údajem. Nutno říci, že toto nahrazení není při změně verze přepsáno, protože pro CVS vlastně zmizí klíčové slovo po prvním nahrazení. Příklad: `cvs_manual.html,v 1.30 1999/12/31 12:58:15 ponkrac Exp`
- `-ko` — žádná klíčová slova se nenahrazují
- `-kb` — žádná klíčová slova se nenahrazují, a navíc systému CVS říkáme, že se jedná o binární soubor.

Práce ve vícečlenném týmu

Pokud pracujete na projektu, v podstatě to normálně probíhá tak, že provedete v nějakém svém adresáři příkaz `checkout`, kterým si nahrajete aktuální verzi projektu na svůj disk do nějakého pracovního adresáře. To v podstatě udělají všichni, kteří se na projektu podílejí. Mezitím každý z nich pomocí příkazu `commit` svojí práci nahrává zpět do systému CVS. Pokud Vás pracuje na projektu více, potřebujete se občas přesvědčit, co se na projektu změnilo, zatímco pracujete. Pokud se chcete pouze přesvědčit, jedno z řešení je použít příkaz `status`, který porovná soubory ve vašem pracovním adresáři s nejnovější verzí uloženou v systému CVS:

```
cvs status
```

Pokud si příkaz vyzkoušíte, program mezi spoustou jiných informací vypíše ke každému souboru File: plus stav vaší pracovní verze. Tento stav je jeden z následujících:

- `Up-to-date` — soubor v pracovním adresáři je shodný s nejnovější verzi v CVS.
- `Locally Modified` — soubor v pracovním adresáři je novější, než verze v CVS. To značí, že jste na tomto souboru prováděl změny, a bude potřeba je později nahrát do CVS příkazem `commit`.
- `Needing Patch` — soubor v systému CVS je novější, než v pracovním adresáři. Znamená to, že jste na tomto souboru nepracoval, ale mezitím někdo jiný nahrál novější verzi do CVS. Pokud potřebujete nutně tento soubor pro svoji práci na projektu, měl byste si stáhnout novější verzi pomocí příkazu `update`, a nebo natáhnout vše znovu pomocí `checkout`.
- `Need Merge` — došlo ke konfliktu. Zatímco jste pracoval na souboru, pracoval na něm též někdo jiný, a mezitím nahrál tuto verzi do CVS. Ale i takovéto konflikty umí CVS řešit, bude potřeba změny sloučit pomocí příkazu `merge`.

Takto to vypadá jednoduše, problémem ale je, že příkaz `status` je trochu „ukecanější“, takže najít tam výše uvedené informace není až tak jednoduché, nebo pohodlné. Téměř ideální řešení nabízí unixová utilita `grep`, která dokáže vyřešit problém velice elegantně a odfiltrovat nepotřebný text. Ve Windows standardně neexistuje prostředek, který by byl schopen odfiltrovat nežádoucí informace. Doporučuji si proto někde opatřit verzi utility `grep` pro Windows, pokud používáte tento systém. Bývá buď součástí překladačů firmy Borland, a nebo lze využít GNU verzi, či verzi dodávanou s freewarovým C/C++ překladačem CygWin pro Windows. Pokud se po této utilitě nechcete pídit, napište mi na moji e-mailovou adresu, a já vám tuto utilitku pošlu ve verzi pro Windows.

S utilitou `grep` dostanete velmi přehledný výpis použitím:

```
cvs status | grep File
```

Později popíšu ještě jedno řešení, jak zjistit stav pracovního adresáře, a to ke konci popisu příkazu `update`. Toto druhé řešení používám já při své práci, a dávám mu přednost před příkazem `status`.

Napsal jsem tedy, jak zjistit, zda vaše pracovní kopie sedí s tím, co je nahráno v CVS. Pokud zjistíte, že potřebujete dohrát novější verze, slouží k tomu příkaz `update`:

```
cvs update jmeno_souboru
```

Příkaz `update` dohraje ze systému CVS novější verzi souboru, pokud jí mezitím někdo z pracovního týmu nahrál příkazem `commit` do systému CVS. Jméno souboru v příkazu `update` je možné vynechat, pak se nahrají všechny novější verze souborů v systému CVS do vašeho pracovního adresáře.

Tak se nabízí i trochu drsnější postup, a to vykašlat se na zjišťování stavu pomocí příkazu status, a rovnou natvrdo natáhnout změny z CVS do pracovního adresáře. Protože tvůrcům systému CVS bylo jasné, že existují i takové lidské vlastnosti, jako je lenost, podpořili i tento postup. Proto příkaz `update` garantuje, že nezničí žádnou vaši práci. Jak přesně příkaz `update` postupuje v jednotlivých případech při možných konfliktech, a jak je řeší, aby neztratil ani ten

nejmenší kousek vaší práce na projektu, či práce dalších lidí v týmu, si ukážeme za chvilku.

Pokud použijete příkaz `update`, systém CVS vypisuje pro každý soubor jednu řádku, a to zhruba tak, že vypíše nějaké písmeno, a pak po mezeře jméno souboru. Vypadá to například takto:

```
U soubor.txt
? soubor.bak
A readme.txt
```

Co tedy znamená první písmeno na každém řádku ve výpisu souboru? Tímto nás systém CVS informuje o tom, co vlastně provedl, nebo by se mělo provést. Význam tohoto písmene je možné najít v tomto seznamu:

- `U` — byla nahrána novější verze souboru z CVS
- `P` — byla nahrána novější verze souboru z CVS, ale systém místo toho, aby přehrál celý soubor použil metodu patche, tedy nahrál jenom rozdíly mezi verzemi. Výsledek je stejný, jako v předchozím případě, ale přeneslo se méně dat po síti.
- `A` — nic se nestalo, jenom vás CVS upozorňuje, že jste tento soubor zaregistrovali pomocí příkazu `add`, a ještě jste neprovedli příkaz `commit`. Je tedy potřeba později provést `commit`.
- `R` — nic se nestalo, jenom vás CVS upozorňuje, že jste tento soubor odregistrovali pomocí příkazu `remove`, a ještě jste neprovedli příkaz commit. Je tedy potřeba později provést `commit`.
- `M` — došlo ke konfliktu, zatímco jste pracovali na souboru, pracoval na něm také někdo jiný, a mezitím nahrál tuto verzi do CVS. Systému CVS se podařilo oba soubory spojit dohromady, a to tak, že prostě do verze z CVS přidal řádky, které máte navíc ve vaší verzi. Navíc vaší původní verzi souboru zazálohoval do souboru s názvem .# plus původní jméno souboru plus tečka a číslo verze vašeho souboru. Doporučuje se prohlédnout obě verze, dát soubor do pořádku a nahrát do CVS příkazem `commit`. Druhá varianta je, že zjistíte, že buď vy, a nebo ten druhý tam psal nesmysly, a jednu verzi prostě smažete a druhou nahrajete pomocí příkazu `commit` do CVS.
- `C` — došlo ke konfliktu, zatímco jste pracoval na souboru, pracoval na něm též někdo jiný, a mezitím nahrál tuto verzi do CVS. Prostě stejná situace jako u `M`, i soubor je zazálohován stejně, jenom se liší tím, že CVS nedokázal tak jednoduše spojit obě verze do jedné. Tento případ je podrobněji rozepsán v dalším textu.
- `?` — označuje soubor, který máte v pracovním adresáři, ale není zaregistrován v CVS.

Jak je vidět, příkaz `update` je celkem bezproblémový, pokud se před jménem souboru neobjeví písmeno `C`, případně `M`, které značí konflikt. Znamená to prostě, že více lidí mění stejný soubor. V takovém případě příkaz `update` konfliktní soubor ve vašem pracovním adresáře zazálohuje. Abych byl trochu konkrétnější, předpokládejme, že konfliktní soubor se jmenuje `pokus.c`. Vy ho ve svém pracovním adresáři máte ve verzi 1.4. Mezitím někdo jiný nahrál do CVS další verzi, kterou CVS automaticky označil jako 1.5. Vy jste použil příkaz `update`, který zjistil konflikt. Proto vaši verzi 1.4 přejmenuje na `.#pokus.c.1.4`, a do souboru `pokus.c` nahraje jakousi sloučenou verzi, která obsahuje jak novou verzi z CVS, tak i změny, které jste udělal na souboru vy. Dále mohou nastat dva případy, a to buď jde o variantu `M`, nebo `C`.

V případě varianty `M` systém CVS prostě přidal do verze s CVS řádky, které jsou navíc ve vaší verzi. A je spokojený. Ale mezi námi, vyznat se v tom je někdy nad lidské síly, protože musíte pátrat, třeba příkazem `diff` po tom, co kde spojil. Abych řekl pravdu, jde prostě o to, že se musíte do souboru podívat, dát to do pořádku, a případně vaši novou verzi nahrát do CVS příkazem `commit`. To samé je nutné i v případě varianty `C`, která je ale trochu přehlednější. Ve variantě `C` je totiž jasně vyznačeno, jaký je rozdíl mezi oběma verzemi. Je potom na vás, abyste dal soubor do pořádku, a nahrál zpět pomocí příkazu `commit` do CVS. Pro ilustraci uvádím část souboru `pokus.c` s vyznačenými rozdíly mezi vaší verzí a verzí uloženou v CVS:

```
fprintf(stderr, "No code generated.\n");
<<<<<<< pokus.c
exit(nerr == 0 ? EXIT_SUCCESS : EXIT_FAILURE);
=======
exit(!!nerr);
>>>>>>> 1.5
```

Z výpisu je patrné, kterou část jste měl vy ve své původní verzi pokus.c, a která část je nahraná ve verzi 1.5 ze systému CVS. Vy potom vymažete ty řádky, které tam nepatří, nahrajete do CVS použitím příkazu `commit`, a je to.

Dlužno ovšem říci, že byste se pokud možno měli vyhnout této konfliktní situaci, kdy dva lidé editují stejný soubor. A nebo ji alespoň eliminovat na minimum.

Slíbil jsem také zhruba uprostřed této kapitoly, že ukáži jiný způsob, jak zjistit stav pracovního adresáře, a nepoužít příkaz `status`. Princip je velice jednoduchý, program `cvs` umožňuje přidat prakticky ke každému příkazu volbu `-n`, která vyzkouší příkaz naslepo. Je to takové divadlo, program `cvs` se tváří, že běží, vypisuje vše, co má, ale ve skutečnosti CVS nic nemění, žádnou akci se soubory nikde neprovádí. A co takhle si vyzkoušet příkaz `update` naslepo? To je ono, protože tím nám bude zároveň vypsáno, jaký je stav souborů v pracovním adresáři. Použijeme také volbu `-q`, která trochu krotí cvs ve výpisech, takže dostaneme skutečně jenom to potřebné:

```
cvs -n -q update
```

Aby to nebylo tak jednoduché, jak je vidět ze zápisu, volby `-n`, a `-q` se musí psát před příkaz! Tedy v našem případě před slovo `update`. Příčina je jednoduchá, tyto volby totiž můžete použít pro každý příkaz, a proto se píší před příkaz. A volby, které slouží jenom pro konkrétní příkaz se píší za příkaz.

Číslování verzí, nastavení čísla verze

Pokud jste pokročili až sem, už toho pro vlastní práci znáte poměrně mnoho. Proto se naučíme některá kouzla, která vám umožní nastavit číslo verze.

Pokud se systémem CVS již nějakou dobu děláte, všimli jste si, že CVS automaticky čísluje verze. Začíná na 1.1, po první změně v nějakém souboru označí soubor jako verzi 1.2, po další jako 1.3, atd. CVS prostě pořád zvyšuje poslední číslici ve verzi. CVS také umožňuje nastavit číslo verze. Pokud tedy nastavíte, že soubor má verzi 3.5, potom po změně ji CVS zvýší automaticky na 3.6, po další změně na 3.7, a tak dále. Je také možné nastavit číslo verze třeba

```
                     větev 1.3.2.2.3 -> +--|1.3.2.2.3.1|--|1.3.2.2.3.2|---- atd.
                                        /
       větev 1.3.2 ->   +--|1.3.2.1|--|1.3.2.2|--|1.3.2.3|--|1.3.2.4|---- atd.
                       /
                      /
----|1.1|--|1.2|--|1.3|--|1.4|--|1.5|--|1.6|---- atd.   <- zde je hlavní větev
                           \
                            \
           větev 1.4.4 ->    +--|1.4.4.1|--|1.4.4.2|--|1.4.4.3|---- atd.
```

Výpis č. 6: Větve — práce na několika verzích současně

na 2.3.0.5, jak je to obvyklé třeba ve Windows. A CVS potom dodržuje stejnou logiku, po každé změně zvýší poslední číslo za poslední tečkou. Všechny ostatní číslice nemění. Takže v tomto případě se dočkáme po změnách verze 2.3.0.6, a tak dále. Pokud chcete změnit i číslice předtím, musíte to udělat ručně.

Číslo verze se dá nastavit pomocí příkazu `commit`, a to pokud zároveň použijeme volbu `-r`, za kterou po mezeře přidáme číslo verze. Takže následující příkaz nastaví všechny soubory v projektu na verzi 2.0:

```
cvs commit -r 2.0
```

Po této akci označí všechny soubory v projektu na verzi 2.0, a CVS bude dále zvyšovat při každé změně poslední číslo. Je samozřejmé, že se vám číslování začne trochu „rozcházet“, protože v projektu to obvykle vypadá tak, že některé soubory změníte jednou, některé vícekrát. Soubor, který jste změnili jednou pak bude mít verzi 2.1, po další změně 2.2, a tak dále. Ale CVS vám mění pouze poslední číslici, vy víte, že se všechno týká 2. verze.

Je samozřejmě možné nastavit číslo verze jenom u jednoho souboru, a to tak, že jméno souboru uvedete na konci příkazu `commit` při nastavování čísla verze:

```
cvs commit -r 3.0 jmeno_souboru
```

Je jasné, že pokud budete často přečíslovávat verzi, můžete si v projektu vyrobit i velice slušný zmatek. Proto vám CVS nedovolí nastavit všechno, co si namanete. Především vám nedovolí snížit verzi. Pokud se například pokusíte nastavit číslo verze u nějakého souboru například na 1.5, když samotný soubor je v CVS uložen jako verze 1.10, systém CVS vám to nepovolí. Prostě se vám to nepovede. Je tedy jasné, že verze s vyšším číslem je vždy novější. Nemůže to být jinak, protože CVS vám dovolí pouze zvyšovat číslo verze směrem nahoru, ale nikdy ne směrem dolů.

Snad jenom poznámku, tímto způsobem je možné nastavit jenom verzi se dvěma číslicemi, pokud nemáte založenou tzv. větev. Proto zatím používejte jenom čísla verzí se dvěma číslicemi, jako je např. 1.5.

Větve — práce na několika verzích současně

Při práci na projektu se obvykle postupuje tak, že se založí projekt. Tím začne vývoj první verze. Než bude k dispozici funkční projekt v první verzi, tak mezitím pracovníci uloží do CVS řadu meziverzí. Poté je oficiálně ohlášena první verze. Mezitím začne vývoj na další verzi. Dále se může stát, že z první verze vznikne několik větví. Například častým případem je, že z hotové první verze se začne vyvíjet druhá verze, a zároveň se ještě první verze opravuje, aby se vychytaly poslední drobné chyby. Někdy třeba z první verze vznikne několik produktů. Takto vzniká potřeba spravovat více souběžných verzí. Proto nabízí CVS prostředek nazývaný větve (v manuálu CVS nazývaná jako „branch“).

Větev představuje jakousi novou odnož projektu. Pro ilustraci jaké verze mohou při použití větve vznikat, uvádím takový malý náčrtek (viz Větve — práce na několika verzích současně).

Na náčrtku je celkem jasné, jak vznikají jednotlivé větve. Prostě si kdykoli řeknete, že začnete novou větev, a můžete na ní pracovat. Kdykoli v budoucnu je potom možné větev ukončit, případně spojit s jinou větví a spojit tak dohromady třeba výsledky práce více týmů. Pro projektové manažery, a pro lepší představu všech bude lepší, když si větve představíte jako jakési subprojekty — dílčí podprojekty většího projektu.

Větev můžeme v základě založit dvěma způsoby, buď tak, že za novou větev prohlásíme přesně to, co teď máme v pracovním adresáři:

```
cvs tag -b jmeno_vetve
```

Při tomto způsobu vlastně odstartuje větev, a jako první verze souborů v této větvi budou uloženy verze, které právě máme v pracovním adresáři. Příkaz vyžaduje, abychom větev pojmenovali nějakým jménem.

Druhým způsobem, jak odstartovat novou větev je vytvořit ji jako kopii nějaké verze souborů již uložených v CVS. Tento způsob nevyžaduje, abychom byli v pracovním adresáři projektu, a proto vyžaduje i jméno projektu:

```
cvs rtag -b -r jmeno_znacky\
 jmeno_vetve jmeno_projektu
```

Takže, aby to bylo jasné, někde v projektu musíme mít verzi, kterou jsme si označili nějakým jménem. Potom následuje jméno větve, prostě si novu větev nějak pojmenujeme. A nakonec přidáme jméno projektu, ve kterém teď vytváříme novou větev. Nutno říci, že tato druhá verze příkazu už je trochu větší sousto, než první způsob vytváření větve, a asi si na něj troufnete, až budete mít CVS trochu zažité.

Jak jste asi pochopili, každá větev má své jméno. Dokonce i hlavní větev má své jméno, které jste zadali při založení nového projektu příkazem `import`:

```
cvs import  jmeno_noveho_projektu\
  jmeno_hlavni_vetve  jmeno_prvni_verze
```

Podstatné je, že toto jméno větve budete potřebovat při práci. Dá se také říci, že jméno větve je vlastně jméno subprojektu, na kterém pracujete.

Takže dál už je to jednoduché. Pokud chci pracovat na větvi (tedy subprojektu), vytáhnu si pracovní verzi:

```
cvs checkout -r jmeno_vetve jmeno_projektu
```

Podobně pracuje i příkaz `export`.

Pokud jsem v pracovním adresáři, mohu si upgradovat svojí verzi na jinou větev pomocí příkazu:

```
cvs update -r jmeno_vetve
```

Pokud provedete příkaz `commit`, systém CVS automaticky správně rozezná, na které větvi pracujete, a nahraje vaše změny správně, takže stačí obyčejné `cvs commit`.

Značky — příkaz tag

Teď se trochu rozepíšu o značkách (v originální mluvě manuálů k CVS se nazývají „tagy“). O co jde? Jak vyplývá z předchozího textu, systém CVS si automaticky čísluje verze tím, že zvyšuje automaticky poslední číslici při každé změně. Zároveň vy si můžete u libovolného souboru změnit číslo verze, takže se vám může stát, že nejnovější verze souborů mají nejrůznější čísla verzí. Například projekt může mít pět souborů s tím, že mají následující čísla verzí: 1.1, 2.3, 2.4, 2.5, 3.2. Chcete si pro budoucí účely označkovat právě tuto současnou verzi. Můžete to udělat všelijak, třeba si můžete zapsat datum a čas, a poté si vytáhnout verzi přesně s tímto datem a časem. (Jak vytáhnout starší verzi bude napsáno později.) Jenomže to nepracuje vždy. Ačkoli jsem se o tom zatím podrobněji nerozepisoval, je možné vyvíjet souběžně několik verzí ve stejném čase.

Zkrátka a dobře, nebudu chodit kolem horké kaše, můžete si současnou verzi označkovat, neboli jí nazvat nějakým jménem. K tomu slouží příkaz tag. Například následující příkaz si označkuje pod jménem znacka_1 současnou verzi projektu:

```
cvs tag znacka_1
```

Poměrně užitečná je volba `-l`, která umožňuje označkovat všechny soubory zaregistrované v adresáři, ale nebude označkovávat soubory v podadresářích:

```
cvs tag -l znacka_1
```

Nutno ještě podotknout, že volba `-l` může být použita pro mnoho jiných příkazů, kde má stejnou funkci. Například nechcete-li nahrát do CVS změny celého projektu, ale jenom souborů v aktuálním adresáři, můžete použít:

```
cvs commit -l
```

Ještě užitečnější je, že si můžete vybrat, který soubor budete označkovávat:

```
cvs tag znacka_1 jmeno_souboru
```

Můžete takto postupně označit několik souborů v projektu.

Takže již víte, že můžete označkovávat verze. Tyto značky se objeví prakticky všude. Pokud si například necháte vypsat logovací výpis pomocí příkazu `log`, zjistíte, že se v tomto výpisu jména značek objevují. Je jasné, že značky můžete používat jak často chcete, a můžete si vymyslet klidně stovky různých pojmenovaných značek. Jména značek k určitému souboru zjistíte snadno příkazem `status`:

```
cvs status -v jmeno_souboru
```

Tento příkaz vypíše mimo jiných údajů i řádek s názvem *Existing Tags:*, pod kterým jsou seřazeny všechny značky, které patří k zadanému souboru:

```
Existing Tags:
    znacka_1                (revision: 2.0)
    importovana_verze       (revision: 1.1.1.1)
    ponny                   (branch: 1.1.1)
```

Samotný příklad konce výpisu tedy ukazuje o značkách mnoho. Okomentuji tedy tento výpis. V prvním řádku je psáno, že existuje značka s názvem `znacka_1`, a že takto byla označkována verze 2.0. Další řádky jsou ale mnohem zajímavější. Abych vám usnadnil pochopení, napíšu, že jsem vypsal soubor z projektu ph, který byl předtím založen pomocí následujícího příkazu `import`:

```
cvs import -m "zalozeni projektu"\
 ph ponny importovana_verze
```

A teď tedy přesně vidíte, co znamenají všechna povinná slova u příkazu `import`. V podstatě jsem založil projekt s názvem ph, a o založení jsem učinil komentář „zalozeni projektu“. Za názvem projektu ph následuje jméno větve `ponny`. A posledním slovem je název značky, v našem případě má název `importovana_verze`. Je tedy vidět, že vás CVS již při zakládání projektu donutí vymyslet si jméno značky, které použije pro označkování prvních verzí souborů. To mimo jiné znamená, že můžete kdykoli přesně zjistit, se kterými soubory jste založili projekt.

Poměrně velký počet příkazů programu cvs vám dovoluje použít volbu `-r`, za kterou po mezeře následuje jméno značky. Tím dostáváte mnoho pomůcek pro práci s projektem. Předem ale musím upozornit, že musíte vědět co děláte. To je potřeba pořádně zdůraznit!

Získání starších verzí souborů z projektu

V některých případech se chcete vrátit ke starší verzi projektu, ať už z jakéhokoli důvodu. A nebo ji prostě chcete mít pro zákazníka, který si nezaplatil nejnovější verzi.

Je například možné vytáhnout soubory z dříve označkované verze:

```
cvs checkout -r jmeno_znacky jmeno_projektu
```

Tento příkaz vám vytvoří adresář s názvem ph, a uloží do něj soubory z projektu ph přesně ve verzi, kterou jste si označili značkou se jménem `znacka_1`. Znovu říkám, buďte opatrní, protože pracujete nejspíše na starší verzi! Používejte toto opatrně, zvláště, chcete-li potom změny nahrávat do CVS příkazem `commit`! Pokud se vám takto podaří omylem nahrát starší verze souborů, budete muset podniknout určité kroky na uvedení projektu do původního stavu.

Asi častějším případem bude potřeba prostě získat soubory samotné, nikoli v pracovní verzi pro potřeby vývoje, ale prostě jenom samotné soubory. Potom se hodí příkaz `export`, který se používá stejně jako příkaz `checkout`, ale na rozdíl od něho příkaz `export` nevytváří pracovní verzi, ale jenom adresář se všemi soubory patřící do projektu. Například získat nejnovější verze souborů projektu lze takto:

```
cvs export jmeno_projektu
```

Automaticky se nabízí, jak vytáhnout označkovanou verzi projektu:

```
cvs export -r jmeno_znacky jmeno_projektu
```


Tip: Při založení projektu příkazem `import` udáváte i jméno značky, které je povinné. Takže možná, aniž jste to tušili, jste si automaticky označkovali verzi souborů při založení projektu. Pokud jste například založili projekt třeba takto:

```
cvs import -m "zalozeni projektu"\
 prvniprojekt poc-tym poc-verze
```

Potom jste automaticky označkovali všechny soubory, se kterými jste založili projekt značkou se jménem `poc-verze`. Potom si kdykoli můžete vytáhnout přesně stejné soubory příkazem:

```
cvs export -r poc-verze prvniprojekt
```

Je samozřejmě také možné vytáhnout i verze podle data, třeba takto:

```
cvs export -D "01/01/2000 00:00" jmeno_projektu
```

Tento příkaz vytáhne verzi souborů, které byly v CVS 1. ledna 2000 o půlnoci, abyste mohli zákazníkovi dokázat, že tehdejší verze nemohla způsobit problém Y2K. Datum se zadává v pořadí měsíc, den, rok.

Stavy verzí souborů

Při vytváření projektu a práci na projektu se každý soubor dostává v zásadě do tří možných stavů. Je to stav, kdy ho vytváříte, a nemáte ho ještě prozkoušený. Potom nic nezaručujete. Takový stav nazývejme experimentální. Dále je to stav, kdy jste soubor otestovali, můžete říci, že jste se snažili zbavit se všech možných chyb. To je stav stabilní verze. A potom je to stav, kdy soubor prošel testováním, a je určen k distribuci zákazníkovi.

Tyto údaje můžete také systému CVS sdělovat. Pokud mu to nesdělíte, CVS automaticky všechny verze všech souborů bude označovat jako experimentální. Stav souboru v projektu vám vypíše v každém podrobnějším výpise změn, třeba v logovacím výpisu. Taktéž příkaz `status` vám vypíše stav souboru. Za slovem *State:* vám vypíše jedno ze tří slov:

- `Exp` — experimentální verze
- `Stab` — stabilní verze
- `Rel` — verze určená k distribuci

Pokud chceme nějakou verzi označit jako stabilní, použijeme příkaz:

```
cvs admin -sStab jmeno_souboru
```

Pokud chceme nějakou verzi označit jako určenou k distribuci, použijeme příkaz:

```
cvs admin -sRel jmeno_souboru
```

Pokud například označíte soubor `pokus.c` jako stabilní verzi, nebo určenou k distribuci, platí to pouze pro tuto verzi. Jakmile soubor `pokus.c` změníte, a natáhnete ho do CVS, potom ho systém CVS automaticky označí zase jako experimentální. Víte tedy, která verze je jaká, a zejména pro vedoucího projektu toto dělení verzí přináší velký užitek.

Základní syntaxe příkazů CVS

Tato kapitola trochu podrobněji podává informace o příkazech cvs. Vzhledem k tomu, že v předcházejícím textu je již mnoho informací o různých volbách, tato kapitola podává celkový pohled na příkazy programu cvs.

Základní syntaxe při používání cvs je:

```
cvs  globální_volby  jméno_příkazu\
  volby_příkazu  parametry
```

Ačkoli to vypadá složitě, tak je to jednoduché. Globální volby jsou volby, které můžete (nebo nemusíte) použít pro každý příkaz, a pro každý příkaz znamenají totéž. Potom následuje jméno příkazu, třeba `commit`. Dále můžete (nebo nemusíte) použít volby, které se vážou jenom k tomu příkazu, který jsme teď použili. Dále mohou následovat parametry, nejčastěji je to jméno souboru nebo projektu.

Protože globální volby jsou stejné pro každý příkaz, vypíšeme si zde ty nejčastěji používané:

- `-d adresář_s_daty_CVS` — umožňuje systému CVS říci, kde máte uložená data pro CVS. Pokud jste si nastavili proměnnou CVSROOT, nebudete tuto volbu potřebovat, jinak musíte tuto volbu použít.
- `-e editor` — umožňuje říci, že pro napsání komentáře si přejete spustit editor, který se vám líbí. Místo slova editor zadejte cestu k vašemu oblíbenému editoru. Pokud tuto volbu nepoužijete, ve Windows se spustí Notepad. Abyste nemuseli svůj oblíbený editor zadávat při každém příkazu pomocí volby `-e`, můžete přidat do AUTOEXEC.BAT (ve Windows, Unixáci do `.profile`) nastavení proměnné CVSEDIT, případně EDITOR.
- `-n` — toto je jakési divadlo. Pokud použijete volbu `-n`, můžete si být jistí, že se nic neprovede, pouze program cvs vypíše na obrazovku to samé, jako by se příkaz provedl.
- `-q` — příkaz omezuje množství zpráv na obrazovku. Řekněme, že toho vypíše o trochu méně, a vypisuje jenom podstatnější věci.
- `-Q` — dělá z cvs neviditelného nindžu. Prostě se program skutečně krotí, a nevypisuje opravdu nic, pokud nenastanou nějaké vážné chyby. Pokud vše probíhá v pořádku, je program tichý, respektive s čistou obrazovkou.
- `-t` — vypisuje naopak co nejvíce na obrazovku. Je vhodné používat společně s volbou `-n`, pokud chcete vyzkoumat, co určitý příkaz přesně dělá.

Pro odvážné — mazání verzí

Někdy se stává, že vám dochází místo na disku. S každou zaznamenanou změnou vám narůstá velikost prostoru zabraného daty systému CVS. Jak bylo již psáno, každá změna se zaznamenává na věky věků. Pokud jste si jisti, že se již nikdy nebudete potřebovat vrátit k některým verzím, lze je ze systému CVS odmazat, a ušetřit tak něco málo diskového prostoru. Používání této možnosti záleží na vašem stylu práce a také na vašich nervech.

Osobně doporučuji ukládat do CVS verze, které jsou alespoň částečně odladěné. Pokud pracujete v týmu, je nutné, abyste do CVS nahrávali pouze funkční verze, aby vaši spolupracovníci mohli na vašich verzích stavět. Pokud jste ale sám a dáváte přednost tomu, ukládat svoji práci do CVS třeba i vícekrát denně, zejména u krátkodobých projektů, může se vám hodit možnost některé nevýznamné

verze později odmazat. Ale pokud chcete znát můj názor, funkce pro odmazávání verzí raději nepoužívám, je možno jimi nadělat velký zmatek.

Pro používání funkce výmazu verzí je potřeba tedy především myslet. Musíte si být absolutně jistí tím, co děláte. Základním příkazem pro odmazání verzí je příkaz:

```
cvs admin -o číslo_verze jméno_souboru
```

Pokud chcete odmazat verzi 1.4 souboru `budulinek.txt`, provedete to takto:

```
cvs admin -o 1.4 budulinek.txt
```

Užitečnější je odmazat verze v určitém rozsahu, například od verze 1.3 do verze 2.5. To se provede takto:

```
cvs admin -o 1.3::2.5 budulinek.txt
```

Tento příkaz dělá přesně to, že nechá verzi 1.3 a nižší, stejně tak jako verze 2.5 a vyšší, a všechny verze mezi tím odmaže.

Závěr

Systém CVS toho umí mnohem více, než jsem zde popsal. Pokud vás něco o zajímá, můžete mi napsat. Také budu rád za veškeré vaše zkušenosti s CVS. Pokud se k tomu dostanu, rád bych tento manuál ještě rozšířil, takže mi napište (4) i tehdy, pokud chcete mít k dispozici případnou další verzi.

Poděkování

Rád bych zde v této kapitole poděkoval všem, kteří mi pomáhali při opravách tohoto manuálu, a jeho zkvalitňování. Konkrétně bych chtěl poděkovat tomuto člověku: Marianu Foltýnovi za velkou pomoc při odstraňování chyb v tomto manuálu. ■

1 CVS download
http://cvshome.org/downloads.html
2 Domovská stránka CVS
http://www.cvshome.org
3 FTP archív CVS
ftp://ftp.cvshome.org/pub
4 Miloslav Ponkrác
mailto:miloslav.ponkrac@interval.cz

Linux — zadarmo a legálně!

CZLUG

Vážení příznivci Linuxu,

rozhodli jsme se, ze je třeba dát najevo, že se nám nelíbí postup Microsoftu a BSA, kdy dobrou věc (oprávněný požadavek na používání legálního software) zneužívají pro své reklamní účely způsobem mnohdy hodně nehorázným. Vytvořili jsme proto webovou stránku (1), která vysvětluje, že existují alternativy k Microsoftu. Obsah této stránky si nyní můžete přečíst také v tomoto příspěvku.

Kampaň Microsoftu a BSA

Mnoho uživatelů počítačů si jistě všimlo reklamní akce Microsoftu „nekupujte nahý počítač“, „tatínku, co to je software“ a dále reklamních praktik tzv. BSA (Business Software Alliance): například vyhrožování a rozesílání videokazet sporného obsahu. Hlavní myšlenkou těchto kampaní je upozornění na nezákonnost používání nelegálního software (s tím se dá jistě souhlasit) a dále se mlčky předpokládá, že náprava takového stavu spočívá v nákupu licencí od firmy Microsoft (s tím se dá jistě nesouhlasit). Forma kampaně BSA (což je v České republice neregistrovaná a tudíž neexistující organizace) je podle našeho názoru na hranici etiky, ne-li za touto hranicí.

„Nahým počítačem“ se v těchto kampaních myslí počítač bez operačního systému. Je sice pravda, že bez operačního systému je počítač většinou nepoužitelný, ale není pravda, že si musíme kupovat počítač s operačním systémem od Microsoftu. Existuje spousta jiných alternativ operačních systémů. Příklad jedné takové alternativy je uveden níže.

Za krajně podivnou obchodní praktiku Microsoftu považujeme skutečnost, že u mnoha prodejců hardwaru není možné koupit počítač bez operačního systému firmy Microsoft. Pokud tedy chceme použít jiný operační systém na hardwaru od takového prodejce, stejně platíme Microsoftu za jeho systém, který nakonec ani nepoužijeme. Prodej jednoho zboží (hardwaru) je zde vázán na prodej jiného (operační systém od Microsoftu), který k provozu onoho hardwaru není bezpodmínečně nutný a tedy může být prodáván samostatně.

Linux

Naštěstí operační systém a další software od Microsoftu není jediný, který může uživatel počítače použít. Existuje software, který je zdarma šířen po Internetu. Licence takovéhoto softwaru obvykle nejenže umožní uživateli použití a kopírování tohoto softwaru bez omezení, ale zejména uživatel dostane software včetně zdrojových textů ve vyšším programovacím jazyce, má možnost tyto zdrojové texty dále upravovat (nebo si třeba i zaplatit jejich úpravu) a modifikované verze může dále šířit, třeba i za peníze. Takovýto **Free software** (2) (ve smyslu svobodný ale i volný) pak dává uživateli nový druh svobody. Není zde vázán na jednoho dodavatele, není nucen při změně protokolů nebo vnitřních formátů programu kupovat další verze a má všechny prostředky k tomu, aby mohl používaný software upravovat podle svých představ nebo za úpravu zaplatit komukoli jinému (tedy nejen původnímu dodavateli).

Software, šířený volně i se zdrojovými texty - někdy také souhrnně nazývaný **Open Source** (3), dnes pokrývá velmi širokou škálu použití od operačních systémů až třeba po grafické editory.

Ano, i operační systém si lze zadarmo stáhnout z Internetu, zkopírovat od kamaráda nebo koupit za mírný poplatek na CD (s tím, že vám není bráněno v dalším kopírování). Jedním z takto dostupných operačních systémů je i systém **Linux** (4). Jde o operační systém s aplikačním rozhraním kompatibilním se systémy UNIX. Pro tento systém je dostupné velké množství aplikací od serverových (databázové stroje, HTTP servery, aplikační servery a podobně) až po uživatelské (desktopová prostředí, prohlížeče WWW, editory obrázků, sázecí systémy a jiné).

Distribuce operačního systému Linux (5) je možno získat zadarmo na Internetu, například na serveru ftp.linux.cz (6). Tento server je podporován **Českým sdružením uživatelů OS Linux (CZLUG)** (7).

Co tedy dělat?

Nenechte se tedy zmást zastrašovacími kampaněmi. Existují i alternativy k systémům firmy Microsoft. Nakupujte hardware u těch prodejců, kteří vám k němu nebudou nutit žádný operační systém, který pak třeba ani nepoužijete. Zeptejte se, o kolik je počítač levnější bez operačního systému.

Tatínku, maminko, co to je operační systém?

Děti se často upřímně ptají na všechno, o čem slyší kolem sebe. Takto může vypadat odpověď:

Chlapče, operační systém je základní počítačový program, který vdechne život každému počítači. Měl by například umožnit spustit více úloh najednou, ale také třeba dovolit pracovat na jednom počítači v jeden okamžik více uživatelům a využít všechny možnosti počítačové sítě, které zdaleka nekončí jen sdílením disků, tiskáren a prohlížením Internetu. Víš dobře, chlapče, že v naší domácí počítačové síti používáme výhradně Linux a v práci taky pracuji jen na UNIXových systémech. Já vím, proč se ptáš, ty kluku ušatá. Někde jsi četl, že jediný legální operační systém je Microsoft Windows. Myslíš si, že ten Linux doma používáme nelegálně? To přece není pravda. Podívej se například na www.linux.cz (8).

Děti často upřímně věří tomu, co někde přečtou, takže je potřeba jim věnovat trochu času a některé věci jim lépe vysvětlit.

Odkazy

Další informace čtenáři naleznou např. na serverech věnovaných FreeBSD (9), na serveru www.gnu.cz je k dispozici český překlad licence, pod kterou je šířeno jádro Linuxu a mnohé další programy (10), na domovském serveru projektu GNU (11), český překlad některých stránek projektu GNU je k nalezeni na serveru www.gnu.cz (12) a na stránkách Slovenského sdružení uživatelů operačního systému Linux (13). ■

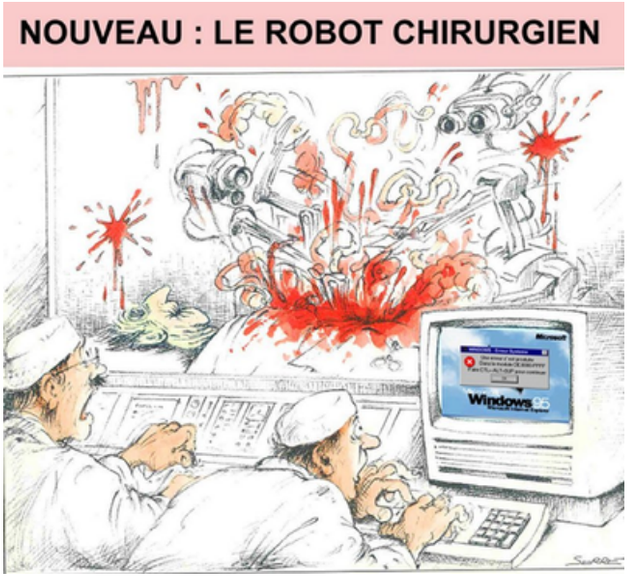
1 Linux - zadarmo a legálně
http://www.linux.cz/zdarma-a-legalne
2 Česká nadace pro podporu free softwaru
http://www.freesoft.cz
3 Open Source Initiative
http://www.opensource.org/
4 Linux.cz
http://www.linux.cz
5 Přehled distribucí Linuxu
http://www.linux.cz/distribuce.html
6 FTP server linux.cz
ftp://ftp.linux.cz/pub/linux/
7 České sdružení uživatelů OS Linux
http://www.linux.cz/czlug/
8 www.linux.cz
http://www.linux.cz
9 Informace o FreeBSD
http://www.freebsd.cz
10 GNU GPL — český překlad licence
http://www.gnu.cz/gplcz.html
11 Projekt GNU
http://www.gnu.org
12 Česká domácí stránka GNU projektu
http://www.gnu.cz/
13 Slovenské sdružení uživatelů OS Linux
http://www.sklug.sk

Zasmáli jsme se!

David Häring

Pokud jste poctivě dočetli až sem :-), tak si zasloužíte nějakou tu relaxaci. Tentokrát jsem vybral mimo jiné pár střípků z linuxové konference (1) a samozřejmě nechybí ani pár kreslených vtipů.

Počítače se stávají každodenní součástí lékařské péče. Ovšem cpát všude počítače za každou cenu nemusí vždycky dobře dopadnout.



Člověk zdaleka není jediným inteligentním tvorem, který se vyznačuje kladným vztahem k výpočetní technice — vezměte si třeba takové myši (rozhovor v konferenci linux):

> … Zní to jako vtip, ale skutečně se to stalo — byla to velmi specifická nečistota na boardu. V počítači se zabydleli 3 brouci a občas krovkami něco zkratovali. Pomohlo vystěhování brouků. U jiného počítače se něco podobného stalo jako důsledek nadměrného množství prachu v počítači. Tyto informace mám od opravářů HW.
> … Tak to jste pěkný béčko -:) Já na vlastní oči viděl počítač, ve kterém si udělali hnízdo myši. Přišlo se na to, až ten úlovek přišel udělat upgrade a počítač byl plný výkalů a na videokartě měly hnízdo z trávy (asi ty video čipy pěkně hřejí). Počítač do poslední chvíle normálně fungoval, asi jsou brouci lepší vodič. Jinak poučení pro všechny, když vyděláte z počítače nějakou kartu, dejte si tu práci a tu díru v krabici zadělejte plechem …

Krátká polemika na téma vzdálené reinstalace systému (opět z konference linux):

> … mám následující dotaz: jak nejlépe provést vzdálenou reinstalaci serveru, když mám přístup pouze TCP/IP a v dohledné době se k serveru nedostanu fyzicky. Původní server běží na Debianu a nový mám připravený na SuSE.
> … Dám jedinou radu podpořenou vlastními neblahými zkušenostmi: nedělejte to. Jakmile něco zvořete (a vždycky se něco pokazí), tak si uříznete pod sebou větev a bude v … onom místě a fyzická návštěva Vás stejně nemine.
> … A jen doplním z vlastní zkušenosti: Už vůbec to nedělejte uprostřed narozeninové oslavy s lahví něčeho dobrého v sobě. Protože pak tam stejně nemůžete jet dříve než vystřízlivíte. :-(

V Linuxu lze nastavit téměř vše, dokladem toho může být třeba následující dotaz:

> Dobrý den.
> <joke> Kde se v Linuxu nastavují práva na psaní písmen 'e' a 'o'? ;-) Neznámým způsobem jsem si tato práva odejmul a nevím, jak je znovu získat. </joke>

Poněkud netradiční návod jak zprovoznit myš pod X Window System (také z konference linux):

> dotaz: nainstaloval jsem RH7 CZ s X-Window (i810, 800*600*24bit). V twm i fvwm2 při prvním pohybu myši (Genius NetScroll+ na psaux) „uteče“ ukazovátko do pravého horního rohu a nemůžu je z tama dostat (maximálně 5 cm po horním kraji)
> odpověď: Máte v levém dolním rohu kočku. Těch 5 cm by tomu odpovídalo. Použijte příkaz kill -9 cat. Pak by to mělo jet. :-)

Tak nějak tučňáci tráví volný čas — aktivním sportem. Někomu to jde lépe, někomu hůře (převzato z www.fudge.cz (2)).

Naopak tady Tux vypadá opravdu poměrně robustně :)

O kulturnosti a „netiquette“ diskusních skupin a konferencí toho bylo napsáno hodně. Některé výrazné znaky (a nešvary) této kultury shrnuje následující příspěvek. Vybráno opět z naší linuxové konference, původní příspěvek byl v angličtině, tady je volněji přeložená česká verze:

> Otázka: Kolik přispěvatelů mailing listu je zapotřebí k výměně žárovky?
> Odpověď: 1343 ?
> – 1 k vlastní výměně žárovky, který napíše do konference, že vyměnil žárovku
> – 14 napíše, že už to někdy dělali a detailně popíšou různé způsoby výměny žárovek
> – 7 upozorní, že výměna žárovek sebou přináší jistá rizika
> – 27 upozorní na pravopisné chyby v mailech pisatelů
> – 53 rozhořčeně reaguje na těch 27, kteří upozornili na chyby v pravopisu
> – 41 se vloží do diskuse o tom, zda je vhodné řešit gramatické chyby
> – 6 napíše svůj názor na to, jestli je správnější dělit slovo žárovka na žá-rovka anebo žárov-ka
> – dalších 6 pošle těch předchozích 6 do <cenzurováno>
> – 156 napíše správci konference o nepatřičnosti diskuse o výměně žárovek v této konferenci
> – 109 napíše do konference, že tato diskuse patří do diskusního fóra „výměna-žárovek“
> – 203 účastníků se ozve s požadavkem, aby příspěvky byly přeposílány též do konferencí „výměna-žárovek“, „kontrola-pravopisu“ a „svítidla-obecně“
> – 111 pošle příspěvky obhajující původní příspěvky, protože žárovky přece používají všichni
> – 306 začne debatovat o tom, který ze zmíněných postupů výměny žárovek je nejlepší, kde se dají zakoupit nejlepší žárovky, která značka žárovek je nejlepší, pro který způsob výměny žárovek, a konečně které značky žárovek jsou špatné
> – 27 pošle URL, kde jsou zdokumentovány různé typy žárovek

– 14 napíše, že URL byly napsány s chybou a pošle opravené URL
– 3 pošlou seznamy odkazů, které jasně ukazují na spojitost problematiky výměny žárovek a tématu konference
– 33 pisatelů ocituje všechny zprávy týkající se žárovek včetně hlaviček mailů, které doposud do konference přišly a požaduje, aby jim diskutující napříště zasílali kopie i na adresu mimo konferenci
– 12 účastníků konference podlehne depresi a z konference se odhlásí
– dalších 19 účastníků se dožaduje Cc: na své adresy
– 4 navrhnou sestavení dokumentu „Výměna-žárovek-FAQ“
– 44 se dotáže, co že to znamená ten „FAQ“
– 4 účastníci tvrdí, že jim to připomíná handrkování v diskusních skupinách Usenetu
– 143 se zeptá co je to ten „Usenet“

■

1 Linuxová konference
http://www.linux.cz/mailing-list/
2 www.fudge.cz
http://www.fudge.cz

Linuxové noviny a jejich šíření

Pavel Janík

Linuxové noviny vydává České sdružení uživatelů operačního systému Linux (1) pro své příznivce a sympatizanty. Vlastníkem autorských práv k tomuto textu jako celku je Pavel Janík ml. (Pavel.Janik@linux.cz). Autorská práva k jednotlivým článkům zůstávají jejich autorům.

Tento text může být šířen a tištěn bez omezení. Pokud použijete část některého článku zde uveřejněného v jiných dílech, musíte uvést jméno autora a číslo, ve kterém byl článek uveřejněn.

Linuxové noviny jsou otevřeny každému, kdo by chtěl našim čtenářům sdělit něco zajímavého. Příspěvky (ve formátu čistého textu v kódování ISO 8859-2) posílejte na adresu (2). Autor nemá nárok na finanční odměnu a souhlasí s podmínkami uvedenými v tomto odstavci. Vydavatelé si vyhrazují právo rozhodnout, zda Váš příspěvek uveřejní, či nikoli.

Registrované známky použité v tomto textu jsou majetkem jejich vlastníků.

Chtěl bych poděkovat společnosti SuSE CR, s.r.o (3), že podporuje redakci v práci na Linuxových novinách.

Linuxové noviny jsou k dispozici také ve formátu HTML na adrese (4). ■

1 České sdružení uživatelů operačního systému Linux
http://www.linux.cz/czlug
2 Adresa redakce
mailto:noviny@linux.cz
3 SuSE CR, s.r.o.
http://www.suse.cz/
4 Linuxové noviny ve formátu HTML
http://www.linux.cz/noviny

Šéfredaktor: Pavel Janík ml.
mailto:Pavel.Janik@linux.cz
zástupce šéfredaktora: David Häring
mailto:dave@ibp.cz
sazba: Ondřej Koala Vácha
mailto:koala@informatics.muni.cz
jazykové korekce: Bohumil Chalupa
mailto:bochal@met.mff.cuni.cz
překlady: Hanuš Adler
mailto:had@articon.cz